ORSAY
Série A
N° d'ordre

254

THESES

Présentées

A L'UNIVERSITE DE PARIS-SUD

CENTRE D'ORSAY

Pour obtenir

LE TITRE DE DOCTEUR INGENIEUR

Par

Monsieur ZIDI Noureddine

1ère THESE: Etude et réalisation d'un système d'entrées-sorties généralisé pour les systèmes de gestion de bases de données.

2ème THESE: Propositions données.

Soutenues le 27 JUIN 1975 devant la Commission d'examen

MM. DEBRAINE Président

GIBERT

DONADIEU Examinateurs

ORSAY
Série A
N° d'ordre

254

THESES

Présentées

A L'UNIVERSITE DE PARIS-SUD

CENTRE D'ORSAY

Pour obtenir

LE TITRE DE DOCTEUR INGENIEUR

Par

Monsieur ZIDI Noureddine

1ère THESE: Etude et réalisation d'un système d'entrées-sorties généralisé pour les systèmes de gestion de bases de données.

2ème THESE: Propositions données.

Soutenues le 27 JUIN 1975 devant la Commission d'examen

MM. DEBRAINE — Président

GIBERT

DONADIEU — Examinateurs

UNIV.
LEIDEN
BIBL.

Ce travail a été effectué sous la direction de Monsieur P. DEBRAINE, Professeur à l'Université de PARIS-SUD, Professeur à l'Institut National des Sciences et Techniques Nucléaires, Adjoint au Directeur du Centre d'Etudes Nucléaires de Saclay.

Je lui adresse mes plus vifs remerciements et lui exprime ma profonde reconnaissance pour avoir bien voulu me faire l'honneur de présider le jury.

Je tiens à remercier Monsieur L. DONADIEU, Directeur de l'Institut Universitaire de Technologie d'ORSAY qui, malgré de nombreuses occupations, a bien voulu accepté d'être membre du jury.

Très conscient de ce que je dois à Monsieur A. GIBERT, Professeur à l'Ecole Nationale des Ponts et Chaussées et Professeur à l'Ecole Nationale des Travaux Publics, qui m'a constamment aidé, qu'il trouve ici l'expression de ma profonde gratitude.

Je tiens également à remercier Madame Marie-Pierre TOLMAIS qui a permis la réalisation et la publication de ce travail dans de bonnes conditions.

TABLE DES MATIERES


INTRODUCTION 1

I- PREMIERE PARTIE

I-A- Les Systèmes de Gestion de Bases de Données (S.G.B.D.) 5

I-A-1 Les systèmes classiques 5
I-A-2 Les bases de données 7
I-A-3 Les S.G.B.D. 10

I-B- Le Système GRISBI 11

I-B-1 La base de données 11
I-B-2 Le système 12
I-B-3 Les machines à accès virtuel et à accès réel 14
I-B-4 Les spécifications externes de GRISBI 15
I-B-4.1. Les structures manipulées 15
I-B-4.1.1 Structure des articles de base 15
I-B-4.1.2 Relations entre enregistrements différents 15
I-B-4.1.3 Structure de la base de données 16
I-B-4.2. Les langages 18
I-B-4.2.1 Les commandes de définition 18
I-B-4.2.2 Les commandes de création 18
I-B-4.2.3 Les commandes d'interrogation et de mise à jour 19
I-B-5 Les spécifications internes de GRISBI 20
I-B-5.1 Mode de fonctionnement 20
I-B-5.2 Configuration à l'installation 21

I-C- Critiques des S.G.B.D. tels que GRISBI 22

I-C-1 Méthode d'accès intégrée 22
I-C-2 Gestion de la mémoire 22
I-C-2.1 Gestion de la mémoire centrale 22
I-C-2.2 Gestion de la mémoire secondaire 24
I-C-3 Universalité des S.G.B.D. 24

II- DEUXIEME PARTIE


II-A- Environnement du système d'E/S généralisé (S.E.S.G.) 26

II-A-1 Contexte système de gestion 26

II-A-2 Remarques 27

II-A-3 Spécifications supplémentaires pour les S.G.B.D. 27

II-B- Architecture du S.E.S.G. 31

II-B-1 Les transformateurs 31

II-B-2 Les spécifications externes du S.E.S.G. 33

II-B-2.1 Niveau utilisateur 33

II-B-2.2 Niveau S.G.B.D. 34

II-B-2.3 Niveau S.E.S.G. 38

II-B-3 Composants d'un transformateur 39

II-B-3.1 Les tables 40

II-B-3.2 Les modules 40

II-C- Spécifications internes du S.E.S.G. 41

II-C-1 Concept et descripteurs de mémoire virtuelle 41

II-C-1.1 Descripteurs de la machine de mémoire virtuelle (MMV) 46

II-C-1.2 Mécanismes et algorithmes de gestion de l'espace mémoire réelle du S.E.S.G. 58

II-C-1.2.1 Lecture d'un segment 65

II-C-1.2.1.1 Segment en mémoire 65

II-C-1.2.1.2 Segment manquant 65

II-C-1.2.2 Ecriture d'un segment 67

II-C-2 Verrouillage des segments 68

II-C-3 Tables des contextes 71

II-C-3.1 Table des fichiers 71

II-C-3.2 Table des organisations de fichiers 72

II-C-3.3 Table des méthodes d'accès 74

II-C-3.4 Table des verrous 75

II-C-4 Liaisons entre les tables du S.E.S.G. 76

II-C-5 Etapes de génération des tables et modules 77

II-C-5.1 Génération "SYSTEME" 77

II-C-5.2 Phase d'exécution 79

III- TROISIEME PARTIE


III-A- Généralités : les programmes du S.E.S.G. 81

III-B- Présentation générale des programmes 82

III-C- Organigrammes et listings des programmes 87

III-C-1. Demande de place en M.C. module DEM 87

III-C-2. Restitution de place en M.C. module RESTI 92

III-C-3. Libération de tables et enregistrements: module LIBERE 97

III-C-4. Création de tables du S.E.S.G.: exemple de CRSB 105

III-C-5. Ajout d'un enregistrement dans un fichier: module CREER 115

III-C-6. Choix de la technique d'accès aux données: module DISTRI 132

III-C-7. Exemples de programmes d'E/S 138

III-D- Exemples d'interrogation de bases de données 146

CONCLUSION 165

BIBLIOGRAPHIE 169

# INTRODUCTION

Le but de ce travail a été de concevoir et de mettre au point un logiciel destiné à gérer et à exécuter toutes les entrées-sorties pour les systèmes de gestion de bases de données.
Ce logiciel que nous avons appelé "Système d'Entrées-Sorties Généralisé" (S.E.S.G.) joue le rôle d'interface entre les systèmes de gestion de bases de données et les divers systèmes d'exploitation des constructeurs.

L'idée maîtresse qui a présidé au développement de ce système découle des faits suivants:

1- Les systèmes de gestion de bases de données,souvent très complexes, coûtent très chers au niveau du développement interne ainsi qu'au niveau de la maintenance. Il fallait pouvoir éviter les modifications trop importantes, voire même la reconsidération totale de ces systèmes du fait de l'évolution rapide des logiciels de base et du matériel.

2- Les méthodes d'accès aux données des constructeurs n'ont pas été conçues spécialement pour les systèmes de gestion de bases de données.

Il nous a alors semblé indispensable de rendre les systèmes de gestion de bases de données aussi indépendants que possible du logiciel de base ainsi que du matériel des divers constructeurs.

Suivant cette voie, nous nous proposons de fournir un système d'entrées-sorties généralisé pour les systèmes de gestion de bases de données dont les spécifications sont celles définies par le rapport du CODASYL (Data Base Task Group de l'Association for Computing Machinery Avril 1969 et Avril 1971).

Dans une première partie, nous préciserons les bases fondamentales de notre étude par un rappel des caractéristiques générales des systèmes de gestion de bases de données.
Nous présenterons ensuite le système "GRISBI" ( Gestion Rationnelle des Informations Stockées dans une Base de Données Intégrée ) que nous avions conçu et développé (équipe de trois ingénieurs) au Service des Etudes Techniques des Routes et Autoroutes du Ministère de l'Equipement.
Initialement, ce système a été réalisé en suivant une méthode classique, c'est à dire intégrant les méthodes d'accès dans le système lui-même.
Nous décrirons certaines difficultés importantes qui apparurent lorsque nous voulûmes adapter notre système à la nouvelle méthode d'accès VSAM (Virtual Sequential Access Method).
Le problème est qu'il ne s'agissait pas, finalement, de modifier le système de gestion, mais de trouver une solution, la plus générale possible, afin de pallier ce type de changement. Cette solution doit permettre aux systèmes de gestion de bases de données de profiter de tout apport nouveau, sans bouleversement dans leur conception et leur architecture.

Dans la seconde partie, nous présenterons notre système destiné à assumer toutes les fonctions d'entrées-sorties.
L'isolement de ces fonctions facilite et assure, dans certains cas, une totale transportabilité des systèmes de gestion de bases de données.
Les mécanismes de mise en oeuvre des programmes et des différentes tables du système seront décrits de façon détaillée.

Enfin, certains points,ainsi que des extraits de programmes ont été regroupés dans la troisième partie.

# I- PREMIERE PARTIE

## I-A LES SYSTEMES DE GESTION DE BASES DE DONNEES

## I-B LE SYSTEME GRISBI

## I-C CRITIQUE DES SYSTEMES DE GESTION

## I-A- LES SYSTEMES DE GESTION DE BASES DE DONNEES ( S.G.B.D. )

### I-A-1 LES SYSTEMES CLASSIQUES:

Il serait presque inutile d'insister sur les multiples inconvénients d'une informatique basée sur la création et l'exploitation de fichiers classiques.

Les tares de cette informatique s'expliquent facilement par les points suivants:

- multiplicité de la saisie et du stockage de la même information.
- risque de non concordance de la mise à jour .
- rapprochement difficile d'informations contenues dans des fichiers différents.
- tâches de maintenance excessivement complexes en cas de changement de la nature ou du format des données.
- etc....

Quant aux programmes d'applications, il va sans dire qu'à chaque modification (au niveau du fichier ou au niveau de la structure des données ...) il est obligatoire de porter des corrections dans la déclaration des éléments traités et parfois dans les séquences d'instructions, ce qui entraîne une perturbation de la logique de ces programmes.

Tout ceci nécessite un travail important qui mobilise une grande partie de la ressource humaine affectée aux tâches informatiques.

Ces problèmes restent entiers quel que soit le type d'organisation des fichiers (séquentiel, séquentiel-indexé, direct ..) et quelle que soit la méthode d'accès choisie.

D'autres inconvénients annexes, mais également importants, se greffent aux premiers. Ils sont relatifs à:

- une optimisation difficile de l'accès aux données pour l'ensemble des programmes d'application.
- une lenteur de mise au point de ces programmes due essentiellement aux défectuosités précedemment citées.

C'est pourquoi le souci de supprimer ces inconvénients a engendré , il y a maintenant une dizaine d'années, un concept qui a fait depuis fortune, celui des bases de données.

## I-A-2 LES BASES DE DONNEES:

Le concept de bases de données repose sur une notion d'intégration des données par opposition à une organisation cloisonnée.

Donnons une définition simple:

- Une base de données est une collection centralisée de toutes les données cohérentes, non redondantes, stockées pour les besoins d'une ou plusieurs applications.

Dans cette définition, la notion de liaisons logiques ou physiques reste volontairement implicite.
Notons qu'il existe plusieurs catégories de bases de données de structure interne différente:

- structure hiérarchique simple.
- structure en multilistes hiérarchiques.
- structure en réseau etc .....

Pour mieux comprendre ce concept de bases de données, il est bon de reprendre et d'analyser une idée émise par la "Harward Business Review" en Septembre-Octobre 1973 (Richard L. NOLAN).

Une approche traditionnelle et une approche base de données sont représentées par les deux figures suivantes:

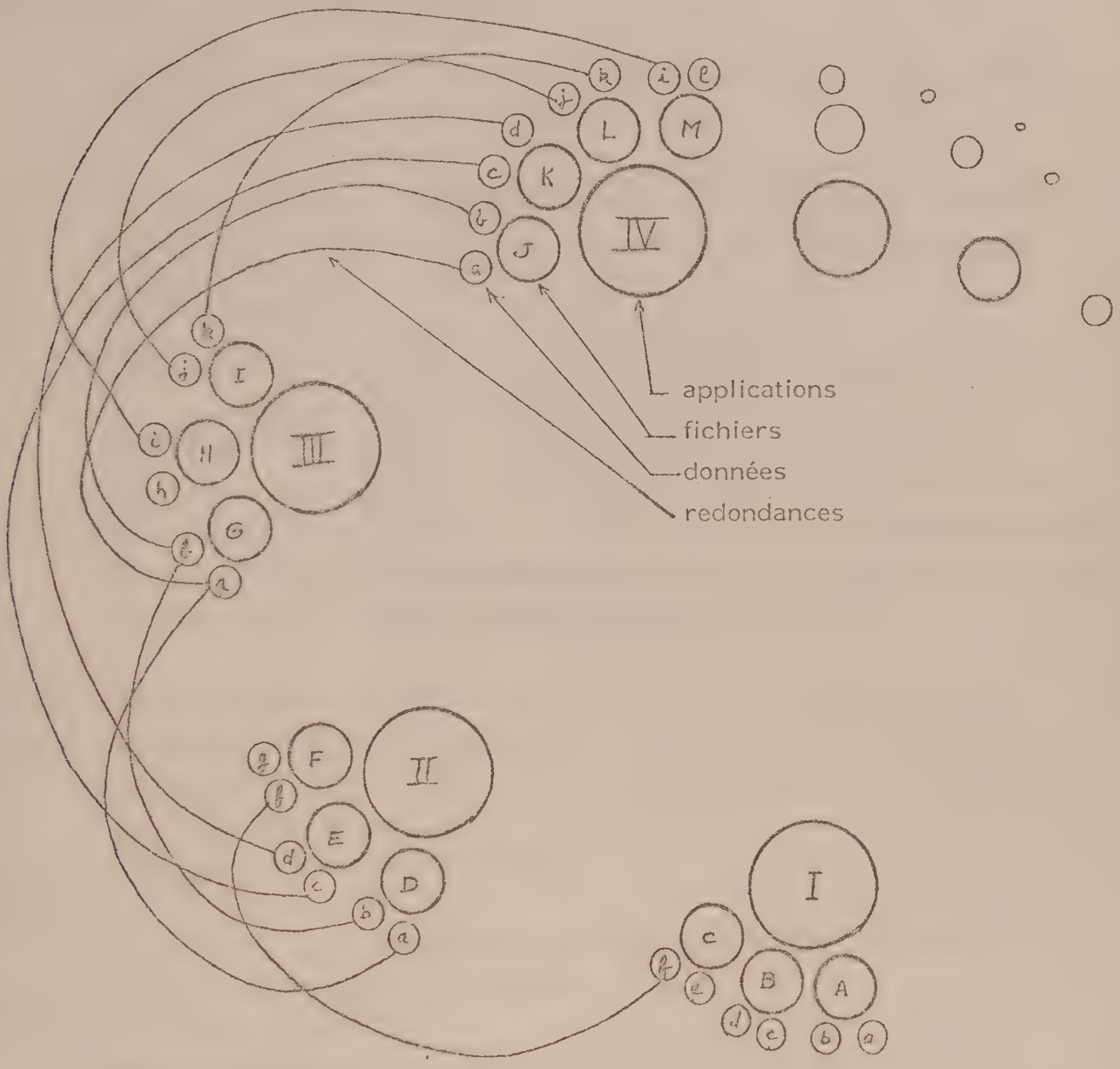



Figure I-1 : Approche traditionnelle

I, II, III, IV, etc ... représentent les applications successives mises en place. Ces applications sont respectivement articulées autour des fichiers A+B+C, D+E+F, G+H+I, J+K+L+M etc ...).

Cette figure met en évidence la redondance des informations (a,b,c,d,e,f etc...) contenues dans des fichiers différents.

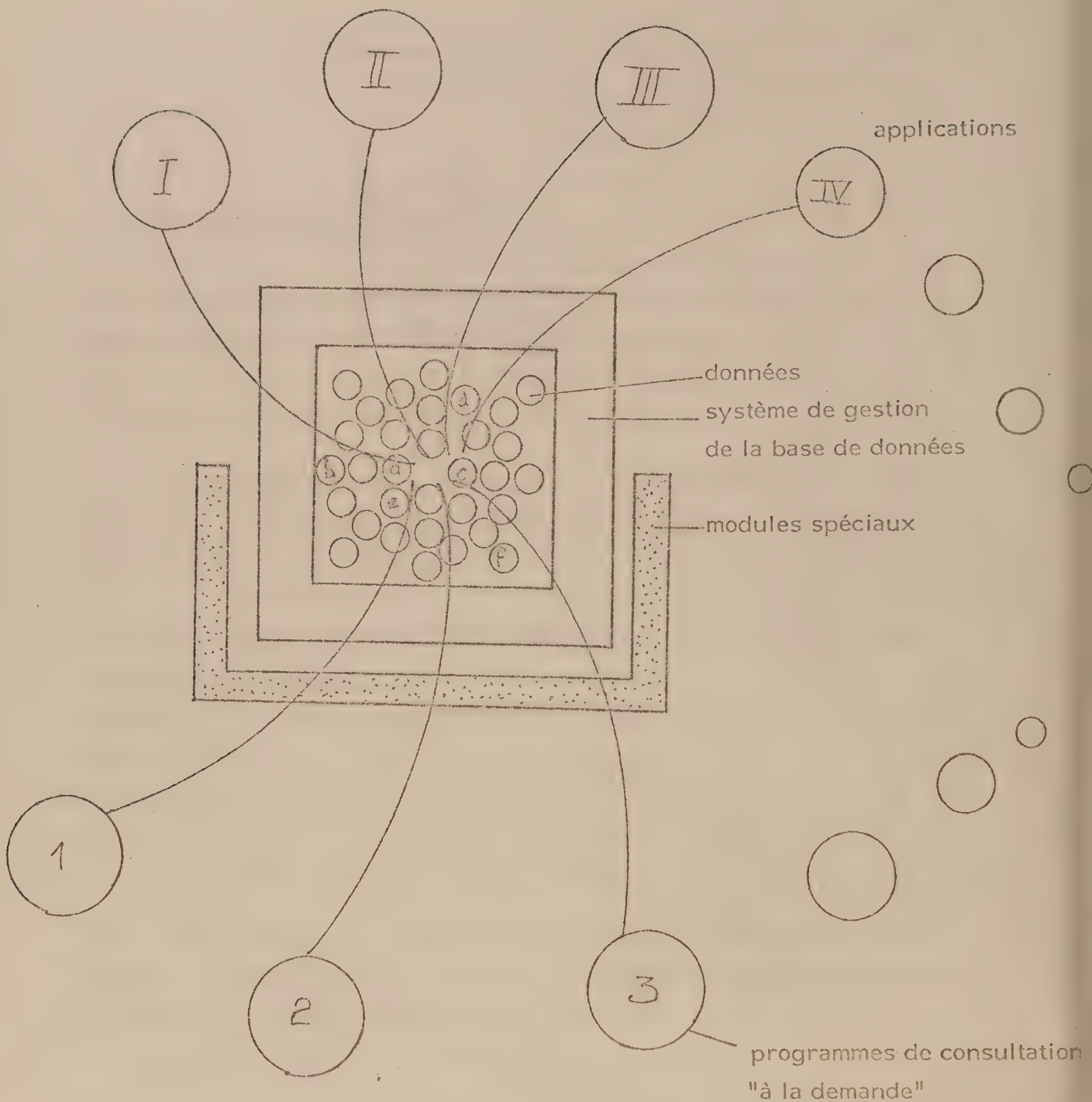



Figure I-2 : Approche base de données.

Les données a,b,c,d,e,etc.. sont uniques et centralisées.
Cette représentation séduisante d'une base de données montre bien qu'il est nécessaire de disposer d'un outil important pour mener à bien toutes les tâches de structuration, de création etc.. de cette base.
Cet outil s'appelle "Système de Gestion de Bases de Données" (S.G.B.D.).

## I-A-3 LES SYSTEMES DE GESTION :

La proportion des installations d'ordinateurs ayant en place un système de gestion de bases de données ne cesse de croître.

Certains organismes, tels que la "Software AG North America", estiment que cette proportion devrait atteindre , dans dix ans, 80 % ( actuellement 6 % environ) .

Même si ce chiffre nous paraît optimiste, il nous montre l'intérêt que les informaticiens portent au problème.

Cependant, la complexité de construction des systèmes de gestion de bases de données devient de plus en plus grande et varie en fonction des exigences des utilisateurs. Elle varie également en fonction des objectifs de plus en plus nombreux que ces systèmes doivent atteindre.

L'expérience des dernières années est là pour prouver que bon nombre des tentatives de mise en oeuvre de ces systèmes de gestion, aussi bien en Europe qu'aux Etats-Unis, ont abouti à des résultats décevants voire même catastrophiques.
Comment expliquer cette situation ?

Pourquoi ces outils sont-ils souvent, très rapidement, en retard sur l'évolution du logiciel et du matériel utilisés ?

Nous pensons apporter, par notre système S.E.S.G., une solution à certaines des nombreuses questions qui se posent.

## I-B- LE SYSTEME GRISBI:

### I-B-1 LA BASE DE DONNEES :

Un certain nombre de fichiers concernant la ROUTE existent au sein du Ministère de l'Aménagement du Territoire, de l'Equipement , du Logement et du Tourisme.

Chaque année, un fichier des accidents corporels de la circulation routière est constitué au SETRA (Service des Etudes Techniques des Routes et Autoroutes) .
L'encombrement d'un fichier annuel est d'environ 80 millions d'octets.

D'autres fichiers sont constitués par les services des Ponts et Chaussées; c'est le cas d'un certain nombre de fichiers de recensement des sections de routes répertoriées, contenant des informations statistiques et techniques.

Il a été envisagé de constituer un fichier "infrastructure" destiné à décrire de façon assez approfondie les caractéristiques du réseau routier national, section de route par section de route.

L'ensemble de ces fichiers constitue alors une base de données routières qu'il faut mettre à jour en permanence et interroger pour les besoins d'études de sécurité, d'investissement etc ...

Le besoin d'un système de gestion de la base de données intégrée a été alors ressenti.

I-B-2 LE SYSTEME:

La diversité des fichiers devant constituer la base et celles des applications à traiter entraînent, pour le système de gestion à mettre en place, certains objectifs tels que:

- la prise en compte d'articles de taille variable munis d'une structure hiérarchique générale, englobant notamment des groupes répétitifs susceptibles d'être définis de façon conditionnelle, un groupe répétitif étant un ensemble d'éléments ou de groupes introduit n fois différentes.

- la possibilité de création de références entre articles de types différents pour supprimer les redondances d'informations et permettre des interrogations croisées.

- la souplesse des langages de définition et de manipulation de la base de données.

- l'obtention, à l'aide de combinaisons logiques, de critères de sélections élémentaires, d'informations ponctuelles et la sortie de résultats statistiques simples (moyenne, écart-type, maximum, minimum, somme, moyennes pondérées etc...).

Cet ensemble d'exigences a permis de développer le système GRISBI suivant l'idée d' E.W. DIJKSTRA, qui consiste à décomposer l'ensemble du système en un certain nombre de modules ou machines de la manière suivante:

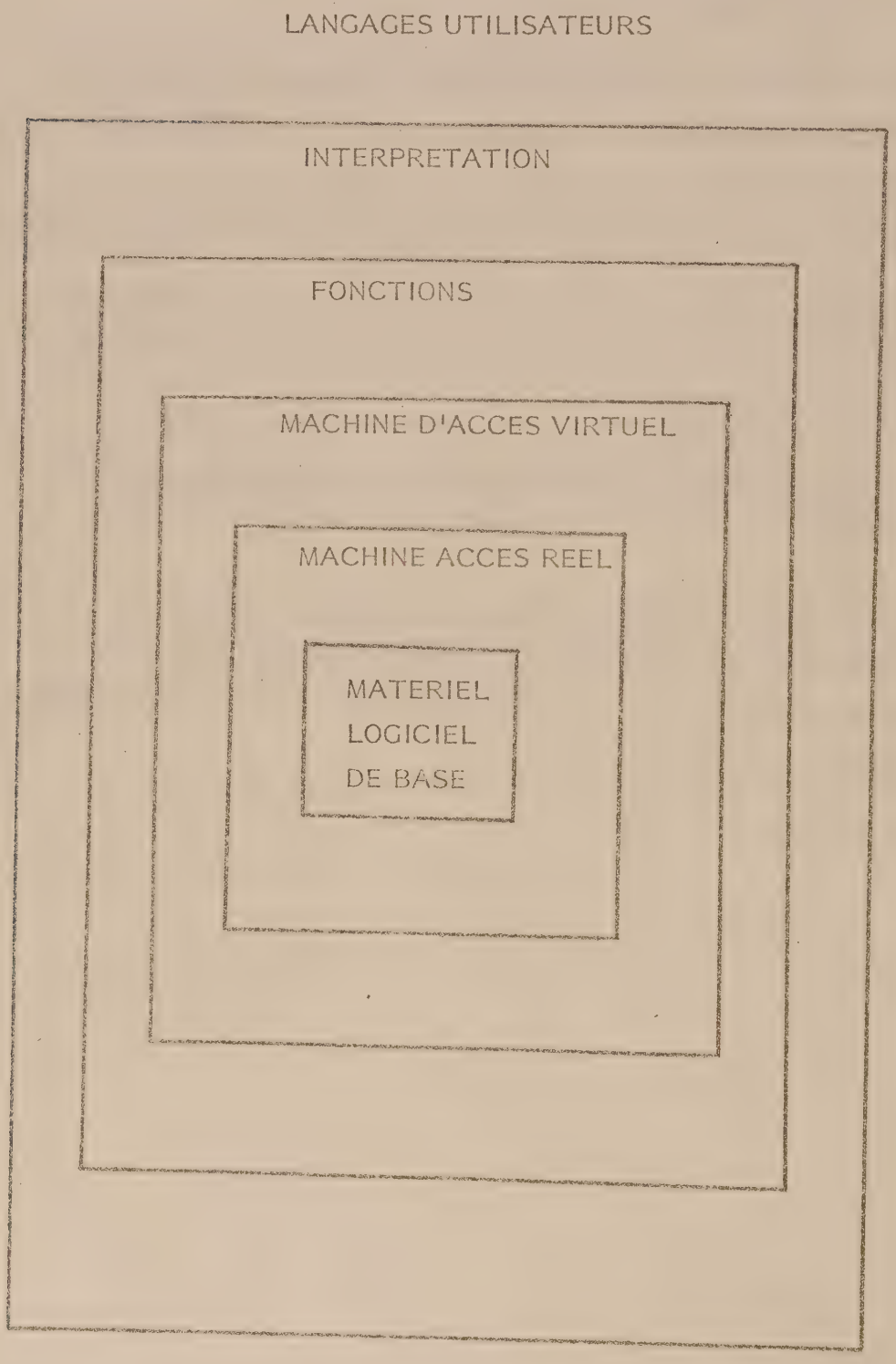



Figure I-3 : découpage du système GRISBI

La figure 3 montre bien:

- que les modules de la machine d'accès virtuel ne font interface qu'entre la machine d'accès réel et la machine d'exécution des fonctions.

- que les modules de la machine d'exécution des fonctions ne font interface qu'entre la machine d'accès virtuel et la machine de compilation etc ...

Cette méthode d' E.W. DIJKSTRA permet ainsi d'écrire et, surtout, de mettre au point, progressivement, les machines du système en simulant, par exemple, les machines adjacentes.

I-B-3 LES MACHINES A ACCES VIRTUEL ET A ACCES REEL :

Ces deux machines ne formaient, pour des raisons de simplicité de mise au point et de délai de réalisation, qu'une seule couche dans la version initiale de GRISBI.

Leur séparation est justifiée dans la seconde partie de cette thèse.

Le contact avec le logiciel et le matériel de base est assuré par notre S.E.S.G..

## I-B-4 SPECIFICATIONS EXTERNES :

### I-B-4.1 LES STRUCTURES MANIPULEES :

#### I-B-4.1.1 STRUCTURES DES ARTICLES DE BASE :

Tout article de base est muni d'une structure hiérarchique composée d'éléments et de groupes:

- les groupes (sommets non terminaux de l'arborescence) sont non répétitifs ou répétitifs.
- le nombre des éléments ou groupes constituant un groupe est inférieur ou égal à 15.
- tout élément ou groupe apparaissant dans l'arborescence peut être conditionnel.

#### I-B-4.1.2. RELATIONS ENTRE ENREGISTREMENTS DIFFERENTS :

Le système GRISBI permet de placer dans un article des éléments d'un type particulier : les "références".
On distingue deux types de référence : les références AVANT et les références ARRIERE.

Une référence AVANT R associe à l'enregistrement (référençant) qui la contient un enregistrement (référencé) de type différent.

Une référence ARRIERE n'est définie en liaison qu'avec une référence AVANT R et associe à tout enregistrement référencé les enregistrements qui font référence au même enregistrement E à l'aide de la référence R.

Exemple : dans une base de données contenant des enregistrements ACCIDENT et des enregistrements SECTION de route on pourra:

. - d'une part placer dans tout enregistrement ACCIDENT une référence AVANT (REFSEC) à l'enregistrement SECTION de route correspondant au lieu de l'accident.

- d'autre part placer dans tout enregistrement SECTION de route une référence ARRIERE (REFAC) à l'ensemble des enregistrements ACCIDENT qui contiennent une référence REFSEC désignant cette section.

Cela permettra de rechercher rapidement et commodément l'ensemble des accidents qui se sont produits sur tel type de section par exemple.

### I-B-4.1.3 STRUCTURE DE LA BASE DE DONNEES :

La base de données est découpée en entités logiques formées par les réalisations d'un même article (articles ACCIDENT, articles SECTION, par exemple).
Nous appellerons sous-bases ces entités.

Chaque sous-base est structurée en une arborescence dont chaque niveau réalise un découpage selon des valeurs ( ou tranches de valeurs) d'un élément.

La figure ci-dessous représente la sous-base ACCIDENT découpée en fonction des éléments DEP (numéro du département) et I (en intersection ou non).

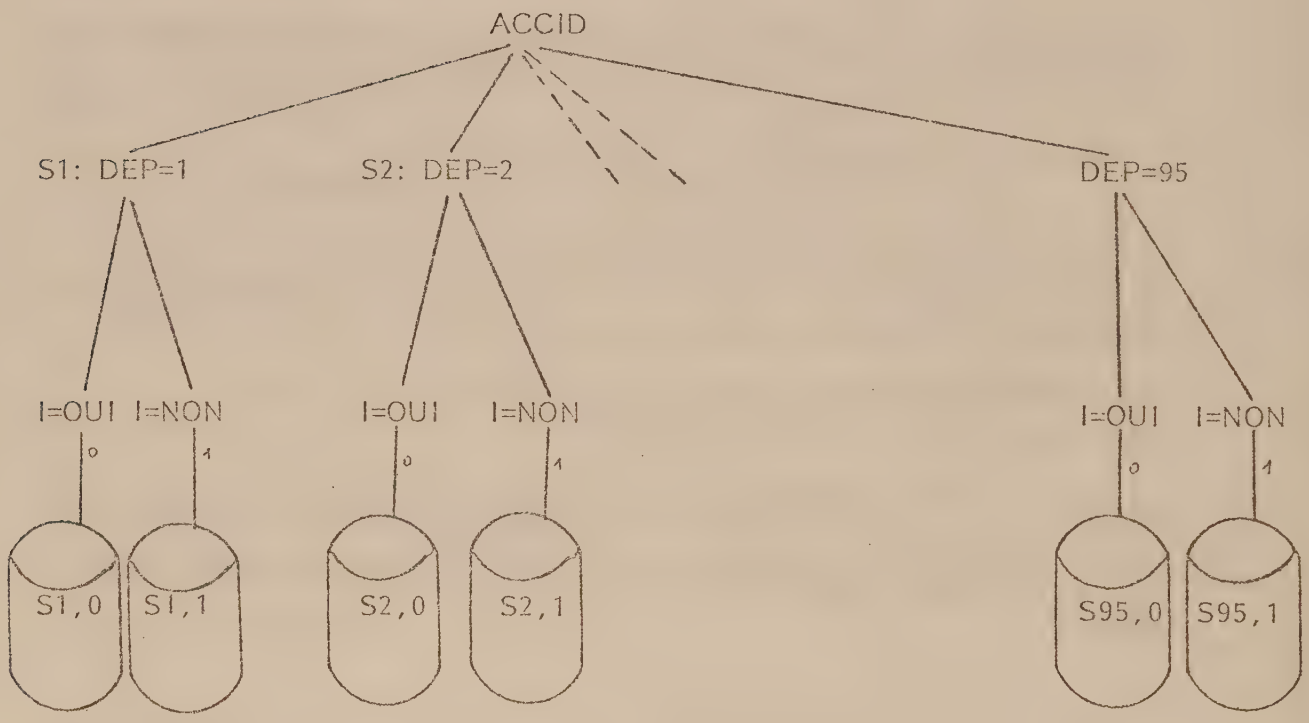



Figure I-4 : structure multiliste hiérarchique d'une sous-base.

Si,j représente le sous-ensemble sous la forme d'une liste d'enregistrements de tous les accidents qui se seront produits dans le département i en intersection (si j = 0).
Une telle structure sera appelée structure multiliste hiérarchique.
Le nombre de niveaux est limité à 8.

Le système GRISBI permet à l'aide d'une commande ( $ DEFSB ) de préciser la définition de cette structure à l'aide d'expressions conditionnelles complexes utilisant les opérations classiques (=, > , < , ≠, >=, < =, OU, ET, ENTRE, RESTE).

### I-B-4.2 LES LANGAGES :

Les langages de définition, de création, d'interrogation et de mise à jour se présentent sous la forme de commandes en format libre.

#### I-B-4.2.1 LES COMMANDES DE DEFINITION :

Elles permettent de définir les articles des fichiers "source" à partir desquels seront crées la base de données ($DEFARTS), ainsi que les articles des sous-bases ($DEFSB).

#### I-B-4.2.2 LES COMMANDES DE CREATION :

Elles permettent de créer une ou plusieurs sous-bases à partir des fichiers "source" pré-définis ($CRESB); elles permettent, d'autre part, d'établir les références "avant" et les références "arrière" relatives à une sous-base ($CREREF).

### I-B-4.2.3 LES COMMANDES D'INTERROGATION ET DE MISE A JOUR :

SADART permet d'ajouter à une sous-base donnée un ou plusieurs enregistrements "base" à partir d'enregistrements placé dans un fichier source donné.

Exemple:

$ADART <nom d'une sous-base> <DD name du fichier source>

Les commandes $SUPART, $SUPREAL, $ADREAL, $MODIFIER permettent respectivement de supprimer un ou plusieurs articles d'une sous-base donnée, de supprimer des réalisations d'un groupe qui vérifie un ensemble de conditions, d'additionner à un groupe G une ou plusieurs réalisations, de modifier les valeurs de certains éléments.
Ces quatre commandes se réfèrent toujours à une commande $RECHERCHER qui permet de définir des ensembles d'enregistrements imbriqués et repérés par des étiquettes.
Enfin, la commande $SORTIR permet d'imprimer suivant un format standard ou autre un fichier de sortie.
Il est inutile de donner le format de toutes ces commandes.
Les exemples de listings donnés en dernière partie suffisent largement pour la compréhension des fonctions de ces commandes.

## I-B-5 SPECIFICATIONS INTERNES :

La connaissance du contenu exact des différentes tables générées par certaines commandes du système GRISBI telles que:

- la T.T. (table triée des noms-dictionnaire)
- la T.S.T. (table de structure des articles)
- la T.O.R. (table des organisations des sous-bases)
- la B.R.T. (bloc de recherche et de traitement)
- la T.C.O.D. (table des codifications)
- la T.O.R.C. (table des conditions des structures des sous-bases)
- etc ...

offre peu d'intérêt pour la compréhension de la suite de notre travail.

Toutefois, nous signalons que l'accès aux données se faisait en PL/1 par la méthode REGIONAL/3 qui est une méthode d'accès direct par numéro relatif de piste et clé sur la piste.
Seule cette méthode d'accès était utilisée.

### I-B-5.1 MODE DE FONCTIONNEMENT DU SYSTEME GRISBI:

Les premières mises au point du système ont été réalisées sur un matériel IBM 360/50 de capacité 512 K octets sous système O.S. M.F.T. (Multiprogramming with Fixed number of Tasks).

La première version opérationnelle (traitement par lots) était disponible depuis la fin de l'année 1973 sur IBM 370/155 de capacité 1536 K octets sous système O.S. M.V.T. (Multiprogramming with Variable number of Tasks), puis sur IBM 370/158 sous O.S./V.S.2.1 (Virtual Storage) permettant d'élargir les possibilités du système opératoire (notamment, support des mécanismes de pagination et de la méthode d'accès VSAM).

### I-B-5.2 CONFIGURATION DE L'INSTALLATION:

La configuration actuelle de l'installation est donnée ci-dessous:

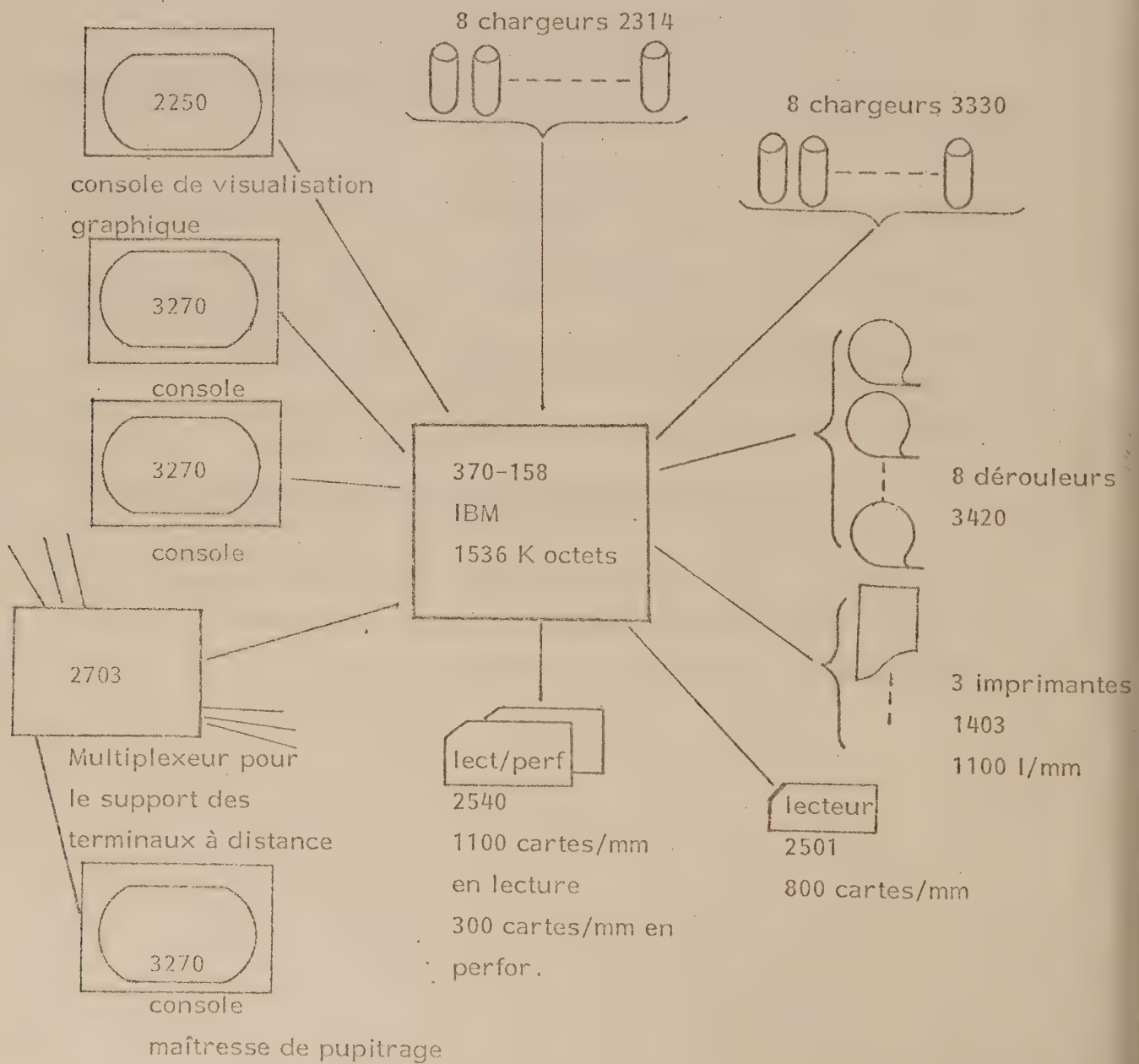



Figure I-5 : configuration de la machine.

## I-C CRITIQUE DES SYSTEMES DE GESTION

### I-C-1 METHODE D'ACCES INTEGREE :

L'intégration de la méthode d'accès REGIONAL/3 dans le corps même du système est néfaste dans la mesure où GRISBI reste dépendant du système opératoire.
Cette technique est classique et la majorité des S.G.B.D. connus, voire la totalité, sont conçus de cette façon.
L'apparition d'une nouvelle méthode d'accès (REGIONAL/x par exemple) plus performante entraînera un bouleversement plus ou moins important dans le logiciel de ces S.G.B.D.

### I-C-2 GESTION DE LA MEMOIRE :

#### I-C-2.1 GESTION DE LA MEMOIRE CENTRALE :

Du fait que toutes les fonctions d'entrées-sorties des S.G.B.D. étaient gérées par les systèmes eux-mêmes, la gestion de l'espace mémoire centrale était réalisée également par ces systèmes.
Cette gestion peut être réalisée suivant deux principes bien différents.

Le premier de ces principes est celui de la gestion de la mémoire réellement disponible dans la partition de ces S.G.B.D..
L'utilisation des macro-instructions GETMAIN et FREEMAIN de l'O.S. - IBM par exemple entraîne une demande ou une libération de place en mémoire centrale.

Ce principe simple réserve parfois des surprises néfastes lorsqu'une demande de bloc mémoire est effectuée pendant l'exécution du système.

Le deuxième principe consiste à définir à l'initialisation des S.G.B.D. une zone en mémoire centrale qui sera utilisée pour toutes les transactions d'entrées-sorties. Les modules de gestion de cette zone sont parfois relativement complexes.

Le problème qui se pose alors est celui de la taille de cette zone d'entrées-sorties.

L'apparition de la méthode d'accès VSAM (Virtual Sequential Access Method) introduit un nouveau concept de mémoire virtuelle. Ce concept très interessant nous permet de définir, au niveau des S.G.B.D., une zone d'entrées-sorties de taille quelconque limitée seulement par la capacité physique de l'A.S.P. (Auxiliary Set Page que nous examinerons ensuite). Cette nouvelle possibilité remet en cause un grand nombre de modules de gestion de l'espace mémoire puisque, à la limite, nous pouvons définir, dans certains cas, une zone capable de recevoir l'ensemble des informations (table et enregistrements) de la base de données.
La gestion de l'espace mémoire centrale se trouve alors considérablement simplifiée.
Par exemple, il suffira, pour allouer une zone de 6 millions d'octets, d'écrire en PL/1 :

```
DCL ZDR (6 000 000) CHARACTER (1) BASED (PZDR);
```

son allocation en mémoire "centrale" se fera par:

```
ALLOCATE ZDR SET (PZDR);
```

Pour que l'adresse soit accessible par toutes les routines S.G.B.D., il est nécessaire de la stocker dans une zone connue par l'ensemble des programmes, soit:

```
DCL SAUV POINTER STATIC EXTERNAL;
```

et au moment de l'initialisation, nous aurons:

```
DCL PZDR POINTER ;
SAUV = PZDR;
```

Ceci met, une nouvelle fois, en relief le manque de souplesse dû à l'intégration de certains modules de gestion des entrées-sorties dans le corps des S.G.B.D.

### I-C-2.2 GESTION DE LA MEMOIRE SECONDAIRE :

Certains S.G.B.D. réalisent une gestion très sophistiquée de l'espace disque en tenant compte des caractéristiques techniques des supports sur lesquels est implantée la base de données.
Le remplacement d'un type de support par un autre entraîne parfois des modifications importantes au niveau des S.G.B.D. (exemple: remplacement des disques IBM 3330 à bras de lecture mobiles par des disques IBM 2305 à têtes fixes etc...).
Notre troisième critique devient alors évidente; elle concerne la séparation des modules de gestion de l'espace mémoire secondaire du corps des S.G.B.D.. Pour les traitements spéciaux, nous proposons une solution dans la deuxième partie.

### I-C-3 UNIVERSALITE DES S.G.B.D. :

Très peu de S.G.B.D. répondent à ce critère d'universalité qui suppose une transportabilité facile de ces systèmes d'un matériel à un autre.
Nous pensons que l'apport de réponses aux trois critiques précédentes ne pourra que faciliter et assurer cette transportabilité.
Certaines restrictions relatives au langage utilisé (CPL 1 par exemple) et à des fonctions spéciales seront examinées.

# II- DEUXIEME PARTIE

## II-A ENVIRONNEMENT DU S.E.S.G.

## II-B ARCHITECTURE DU S.E.S.G.

## II-C SPECIFICATIONS INTERNES DU S.E.S.G.

## II-A ENVIRONNEMENT DU S.E.S.G.

### II-A-1 CONTEXTE SYSTEMES DE GESTION

De nombreuses publications techniques montrent qu'il existe plusieurs types de S.G.B.D. classés parfois différemment d'un auteur à l'autre.

Par notre étude, nous proposons une définition unique d'un "S.G.B.D. modèle" représentatif.

Ce système de gestion de bases de données est constitué d'un ensemble cohérent de programmes permettant d'atteindre la totalité des objectifs définis par les rapports du CODASYL (Avril 1969 et Avril 1971).
Ces objectifs peuvent être résumés de la façon suivante:

- structuration des données en évitant la redondance.
- accès simultané de plusieurs applications aux données.
- existence de plusieurs stratégies de recherches de données.
- protection de la base contre les accès non autorisés.
- regroupement physique des données.
- indépendance des programmes d'applications des données et de leur affectation physique.
- interface base de données et plusieurs langages.
- transparence à l'utilisateur de la manière dont sont réalisées les liaisons entre données.

C'est dans ce contexte que nous avons conçu et réalisé notre logiciel S.E.S.G..
Cependant, certaines remarques s'imposent.

II-A-2 REMARQUES :

Il est impensable de concevoir le S.E.S.G. en espèrant aboutir sur un système permettant de réaliser un interface opérationnel entre n'importe quel système d'exploitation et n'importe quel S.G.B.D. commercialisé ou non avec des spécifications techniques quelconques, même "fantaisistes".

Il est plus raisonnable de se fixer comme objectif final: la fabrication d'un S.E.S.G. permettant de faire interface entre les systèmes d'exploitation d'une manière générale et les S.G.B.D. définis comme précedemment mais satisfaisant en plus à certaines spécifications internes données plus bas.

Toutefois, le S.E.S.G. assurera toujours les échanges entre les systèmes d'exploitation et les S.G.B.D. indiqués.

II-A-3 SPECIFICATIONS SUPPLEMENTAIRES POUR LES S.G.B.D.

1) Les S.G.B.D. étudiés ne possèdent pas de modules faisant corps ou se substituant en partie à certaines fonctions d'un système d'exploitation donné.

2) Les requêtes d'entrées-sorties émises par ces S.G.B.D. à destination du système d'exploitation respectent un certain nombre de règles syntaxiques et sémantiques qui seront définies plus loin. Cependant, le S.E.S.G. de conception évolutive pourra s'adapter, le cas échéant, à de nouvelles requêtes.

3) Chaque S.G.B.D. étudié est écrit en langage évolué employé par de nombreux matériels ou bien il est prévu dans plusieurs types de langages (assembleurs machines).

4) Tous ces S.G.B.D. ne connaissent qu'une méthode d'accès unique. Cette méthode consiste à répartir sur un espace virtuel ("espace adresse") l'ensemble des enregistrements des bases de données, la relation de l'espace adresse avec l'espace mémoire étant assurée par le S.E.S.G..

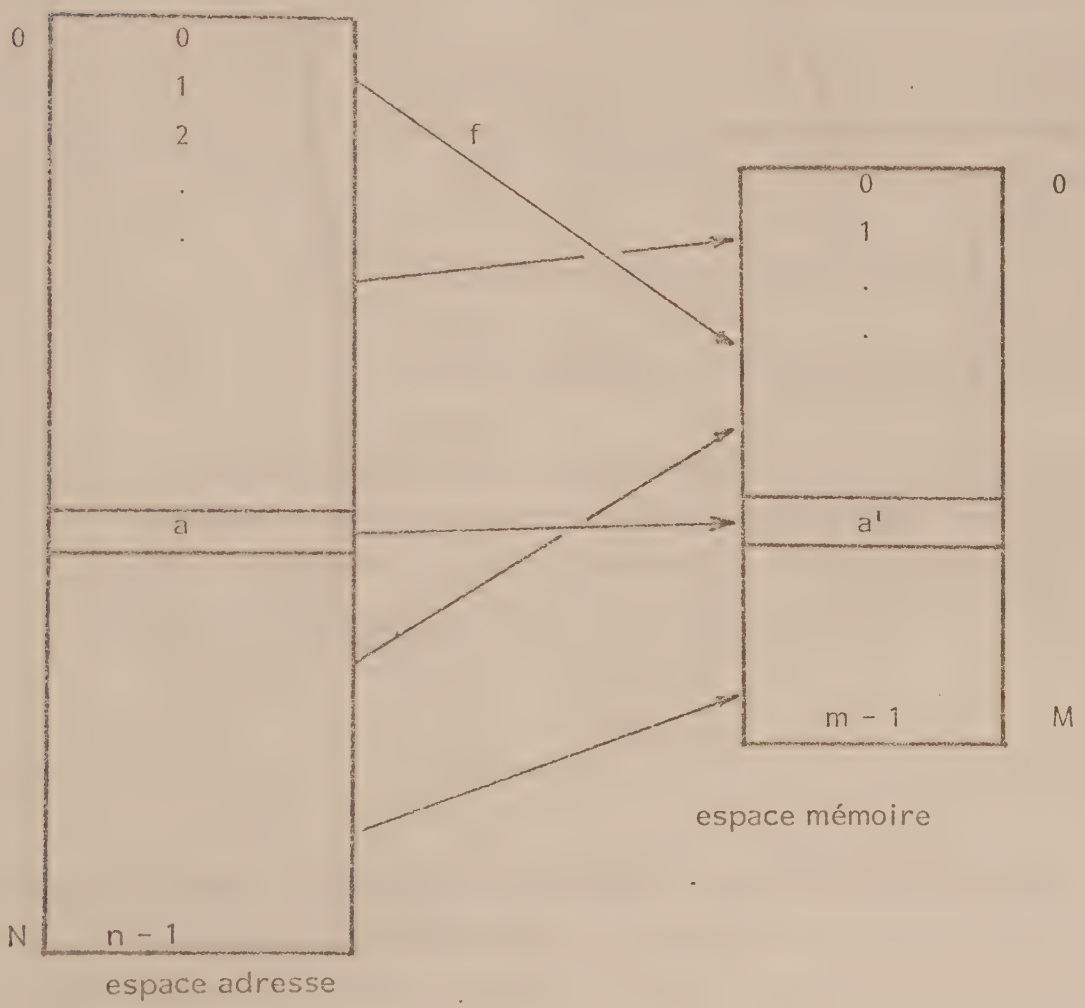



Figure II-1 : Relation "Espace Adresse " ⟷ "Espace Mémoire"

Cette notion de mémoire virtuelle qui a été définie par P.J. DENNING dans "VIRTUAL MEMORY" permet de donner l'illusion, aux S.G.B.D., de disposer d'un espace mémoire centrale plus grand qu'en réalité. En fait, il correspond à la totalité des ressources mémoires utilisables (ressource de mémoire rapide de la partition réservée et ressources supports physiques tels que les disques 3330, 2314 appartenant au système).

P.J. DENNING définit (voir page 158 de "VIRTUAL MEMORY") une fonction f qui, à chaque instant, permet le passage suivant:

$f : N \longrightarrow M \cup \emptyset$ où

l'espace adresse est représenté par l'ensemble N = (0,1,2,..., n-1)
l'espace mémoire est représenté par l'ensemble M = (0,1,2, ..., m-1)
avec $n > m$.

$$f(a) = \begin{cases} a' \text{ si l'élément a est en mémoire en a'} \\ \emptyset \text{ si l'élément a est omis en mémoire} \end{cases}$$

Cette technique est très interessante. Elle permet d'affecter à chaque entité physique (table ou enregistrement) un numéro logique quelle que soit sa position dans l'espace mémoire. Prenons l'exemple des liaisons entre articles du système GRISBI.

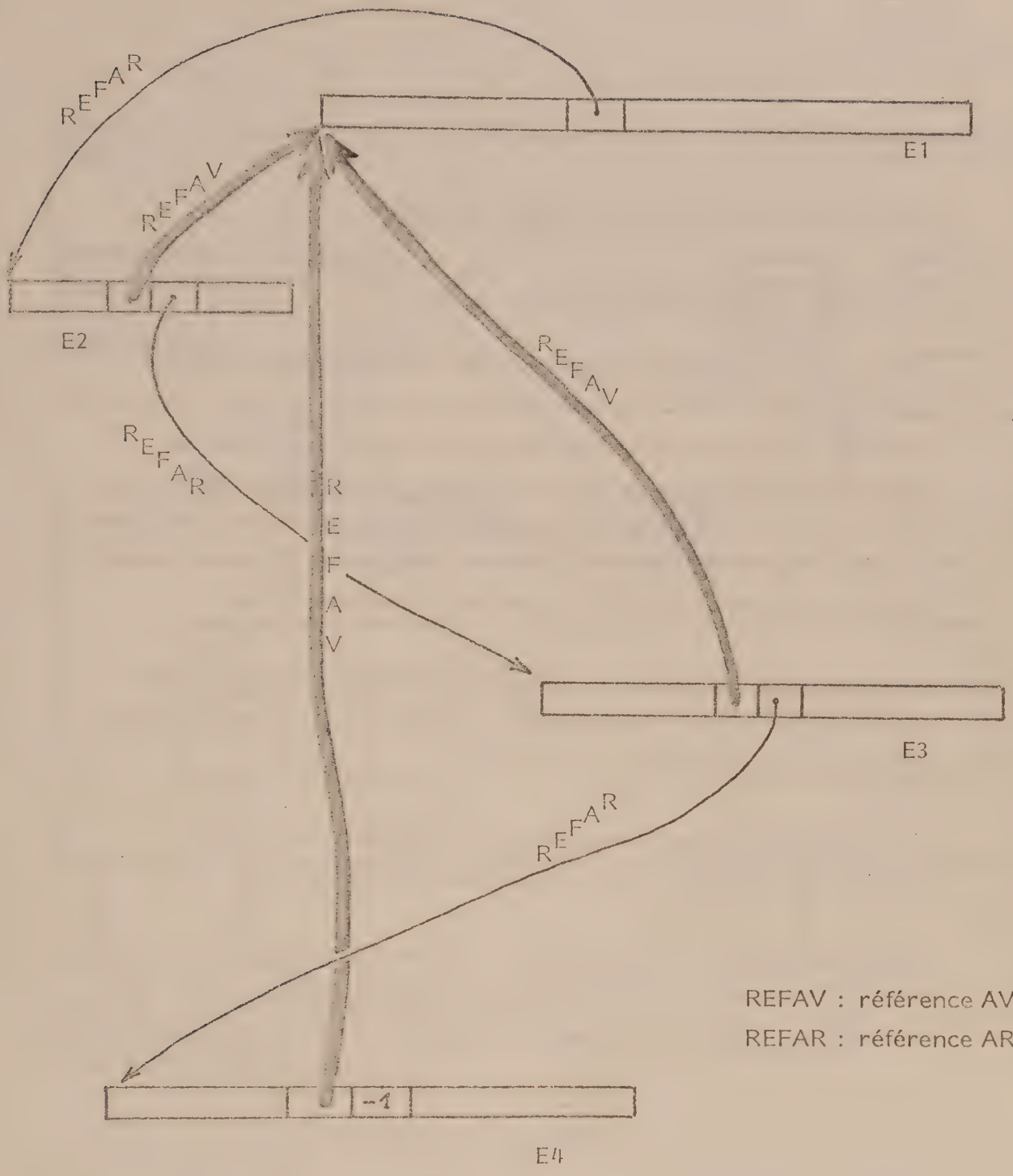



Figure II-2 : liaison entre articles

Toutes les références REFAV et REFAR étant exprimées en adressage virtuel, le déplacement physique de l'enregistrement E1 par S.E.S.G. ne nécessite aucune mise à jour des REFAV des enregistrements E2, E3 et E4.

## II-B ARCHITECTURE DU SYSTEME S.E.S.G.

### II-B-1 LES TRANSFORMATEURS :

Cette notion de transformateurs est liée à la conception modulaire du S.E.S.G..
En effet, notre système assure toujours les échanges entre les systèmes d'exploitation et les S.G.B.D.et, dès que les spécifications internes seront changées ( par exemple, lorsqu'il s'agira d'implanter S.E.S.G. sur de nouveaux S.G.B.D.), il est prévu d'ajouter dans notre système de nouveaux modules et de nouvelles tables.

Les ensembles des tables et modules ainsi définis seront appelés des "transformateurs" qui assurent la prise en charge de nouveaux produits (par exemple).

Chaque transformateur sera associé à une classe de S.G.B.D. ayant les mêmes spécifications techniques internes.

Certaines tables et modules du S.E.S.G. sont communes à l'ensemble des transformateurs. Ils ont été regroupés dans la partie que nous avons appelée TRANSFORMATEUR PRINCIPAL.

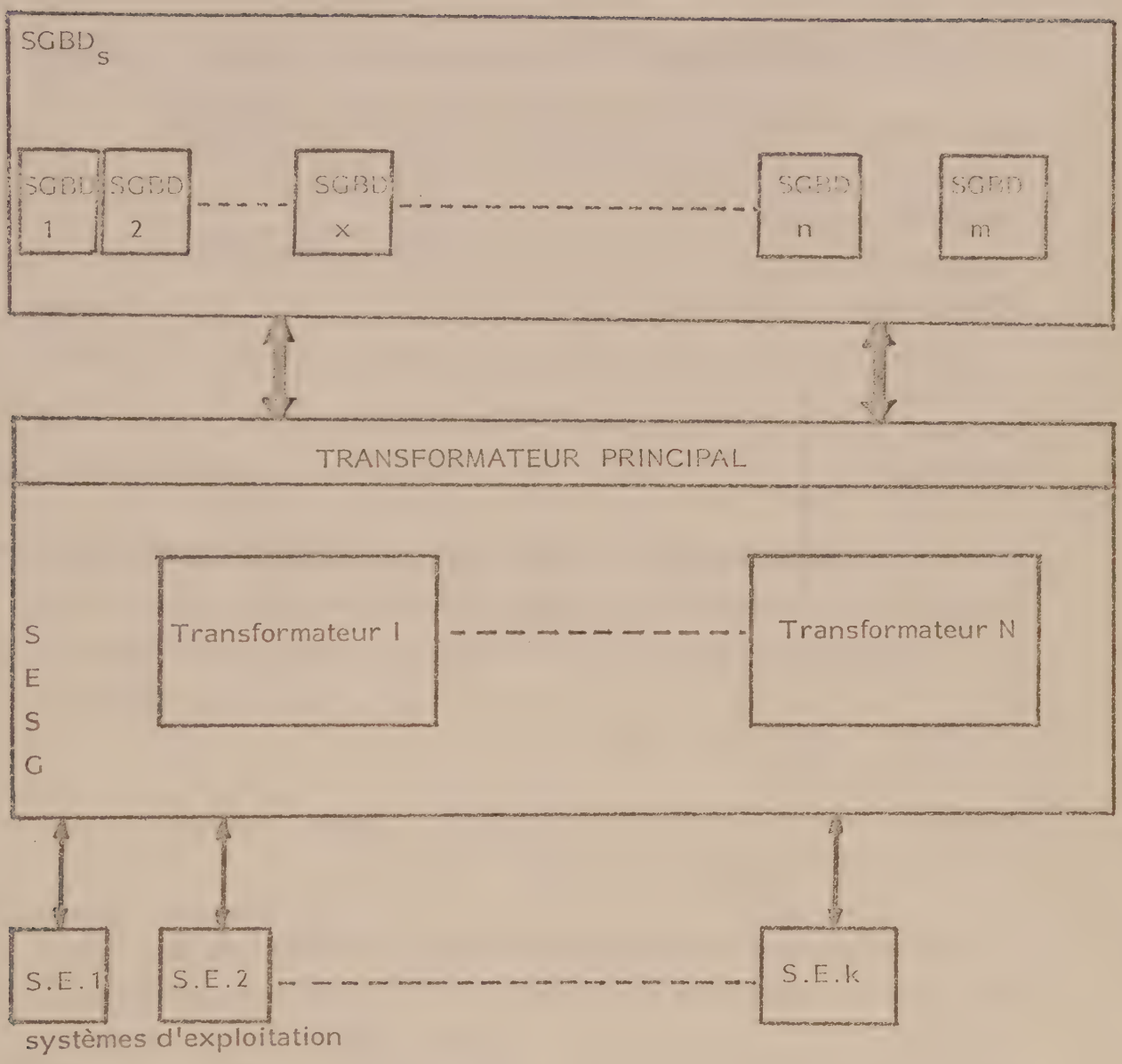



Figure II-3 : architecture générale du S.E.S.G.

Il est clair que sur une installation donnée ayant en place un S.G.B.D. donné, ne seront chargés que les modules du transformateur associé.

## II-B-2 SPECIFICATIONS EXTERNES :

Pour mieux comprendre la signification de chaque spécification technique, nous allons prendre un exemple concret, aussi général que possible.
La figure ci-après indique les quatre niveaux de propagation d'une requête utilisateur:

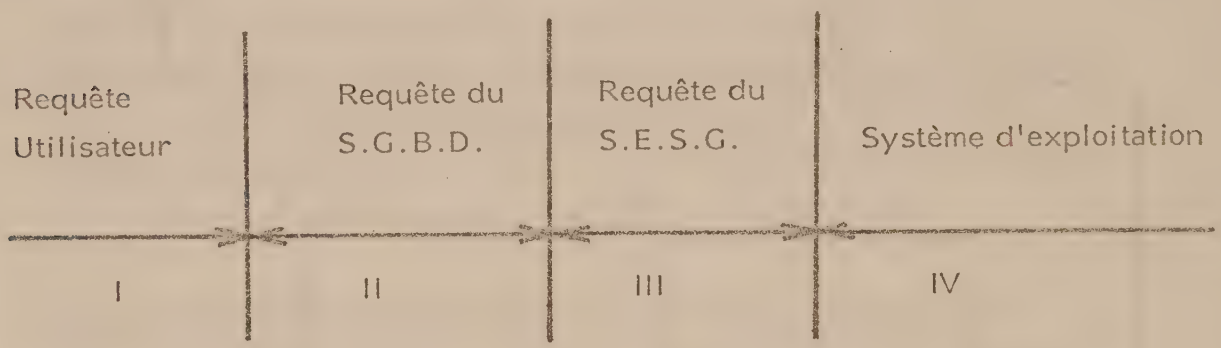



Figure II-4 : niveaux de requêtes.

### II-B-2.1 Au niveau utilisateur :

L'appel au Système de Gestion de Bases de Données se fait par une requête de type :

CALL $SGBD_i$ (FIC, FONC, IDENT, LISTE, ZES1, REP1, PS1, PS2 ...PSN)

Dans le cas de GRISBI, l'appel est réalisé par une commande $COMMANDE de même signification que l'appel ci-dessus.

FIC : nom du fichier sur lequel portera l'opération (nom d'une sous-base dans le cas de GRISBI)
FONC : nom de la fonction que doit réaliser le Système de Gestion de Bases de Données.

IDENT : nom d'une zone contenant l'identification d'un élément ou un ensemble d'éléments de la base de données (du fichier ou de la sous-base).

LISTE : nom d'une zone permettant de donner une liste de champs appartenant à l'élément ou à l'ensemble des éléments concernés par la requête.

ZES1 : nom d'une zone d'E/S de l'utilisateur.

REP1 : nom d'une zone de transmission d'une réponse du $SGBD_i$

PS1, PS2,....PSN : paramètres spécifiques au $SGBD_i$

Cet appel sera soit intercepté par le $SGBD_i$ lui-même, soit par un programme intermédiaire (link-édité avec le programme utilisateur) qui analysera les paramètres et émettra une requête de même type au $SGBD_i$.

II-B-2.2 Au niveau du S.G.B.D. :

Nous pouvons considérer que le S.G.B.D. est une boîte noire faisant interface entre les programmes utilisateurs et le S.E.S.G. Cette boîte noire assure un certain nombre de fonctions schématisées par la figure suivante.

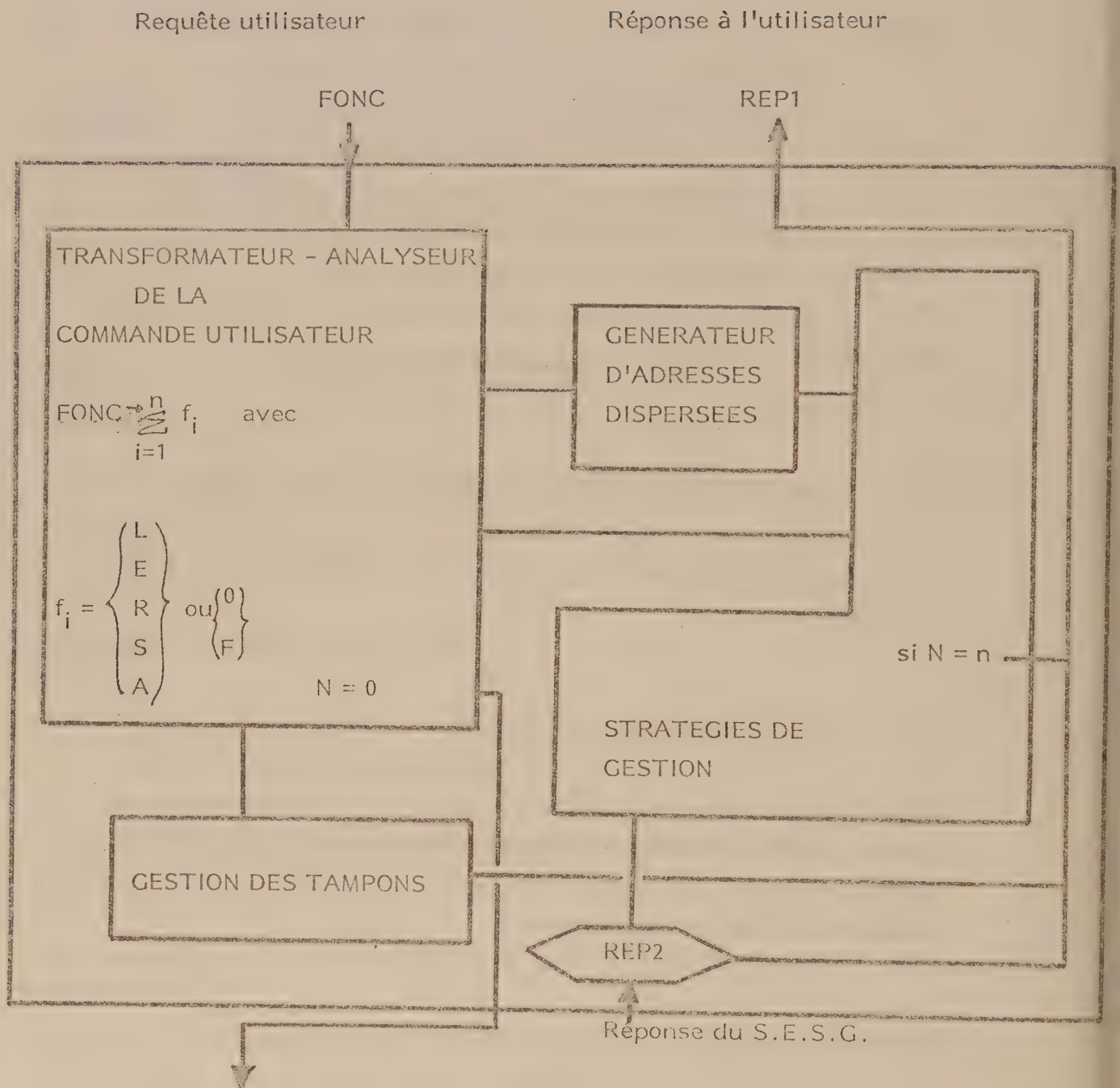



Figure II-5 : S.G.B.D.

Il est évident qu'une requête utilisateur peut porter sur un ensemble d'informations qui peut être très complexe.

Une des fonctions du TRANSFORMATEUR-ANALYSEUR du S.G.B.D. est la décomposition de la fonction FONC de l'utilisateur en une ou n fonctions logiques FONLO du type suivant:

L : lecture
E : écriture
R : remplacement
S : suppression
A : mise en attente

0 : ouverture de fichier ou sous-base
F : fermeture de fichier ou sous-base

(0 et F seront traitées séparément)

Le S.G.B.D. est capable de réaliser cette transformation et gère ses tampons correctement de manière à ne lancer les requêtes au S.E.S.G. que si cela est nécessaire.
Le module ou l'ensemble de modules GENERATEUR D'ADRESSES DISPERSEES permet (dans cet exemple) de transformer le contenu de la zone IDENT en une valeur ou une série de valeurs (adresses virtuelles) interprétable par le S.E.S.G..
Ce module peut être appelé n fois.
Quant au module ou ensemble de modules nommé STRATEGIES, il convient de dire que sa fonction reste très spécifique du S.G.B.D. (par exemple, mise à jour de pointeurs de références au niveau des enregistrements).

Toutefois, quel que soit le S.G.B.D. répondant aux spécifications énumérées dans la partie II-A, les requêtes émises à destination du S.E.S.G. seront du type:

CALL SESG ( NSGBD, NBASE, FIC, FONLO, IDTR, PROX, ORGLO, ZES2, REP2, BCE ) ;

avec :

NSGBD : nom quelconque associé au S.G.B.D.

NBASE : nom de la base de données de travail.

FIC : nom du fichier sur lequel portera l'opération logique.

FONLO : nom de la fonction logique que doit réaliser le S.E.S.G. Cette fonction doit être obligatoirement l'une de ces cinq fonctions L, E, R, S, A.

Les quatres premières fonctions sont les fonctions élémentaires connues. En revanche, la fonction logique A est une fonction que pourra supporter le S.E.S.G. et qui permettra, éventuellement, (si l'option est retenue) d'utiliser notre système dans un environnement de multi-systèmes de gestion de bases de données.
Dans ce cas précis, un S.G.B.D. émettra une requête au S.E.S.G. (requête du type L, E, R, S,) puis, après exécution d'une séquence d'instruction de longueur quelconque, émettra une requête de type A.

IDTR : ce paramètre est relatif à l'identification de l'enregistrement sur lequel portera l'opération logique. C'est une adresse virtuelle dans un espace des adresses N.

ORGLO: ce paramètre permet d'indiquer au S.E.S.G. le type de l'opération. En effet, une lecture séquentielle indexée sur une machine X ne correspond pas obligatoirement à une lecture séquentielle indexée sur une machine Y par exemple.
Le S.E.S.G. doit pouvoir distinguer ce type d'ambiguité.

PROX : ce paramètre spécial est appelé paramètre de proximité.
Dans certains types de S.G.B.D., cette notion tient une place très importante dans la gestion des emplacements des enregistrements de la même chaîne (enregistrements référencés et référençants par exemple).

Ce paramètre contient alors l'adresse virtuelle de l'enregistrement "auprès" duquel (physiquement) le S.E.S.G. doit implanter celui fourni par la requête de type E par exemple:

ZES2 : nom d'une zone d'entrées-sorties du S.G.B.D.$_i$.

REP2 : nom d'une zone de transmission de la réponse du S.E.S.G. au S.G.B.D.$_i$.

BCE : nom d'une zone du S.G.B.D.$_i$ relative à la fonction logique A (attente). Cette zone permettra au S.E.S.G. d'indiquer au S.G.B.D.$_i$ que l'opération demandée est achevée.

II-B-2.3 Au niveau du S.E.S.G. :

Toute requête émise par le S.E.S.G. à destination d'un système d'exploitation donné dépendra :

1) de l'appel réalisé par un S.G.B.D.$_i$ répondant à toutes les spécifications définies.
Un S.G.B.D. donné peut être activé sur des matériels différents; dans tous les cas, il émettra des appels à S.E.S.G. selon la syntaxe décrite plus haut. Seules les valeurs des paramètres varieront.

2) du système d'exploitation lui-même, d'une méthode d'accès du constructeur. Nous donnerons un exemple plus loin.

II-B-3 COMPOSANTS D'UN TRANSFORMATEUR :

La figure II-6 donne un aperçu de l'architecture interne d'un transformateur du S.E.S.G.

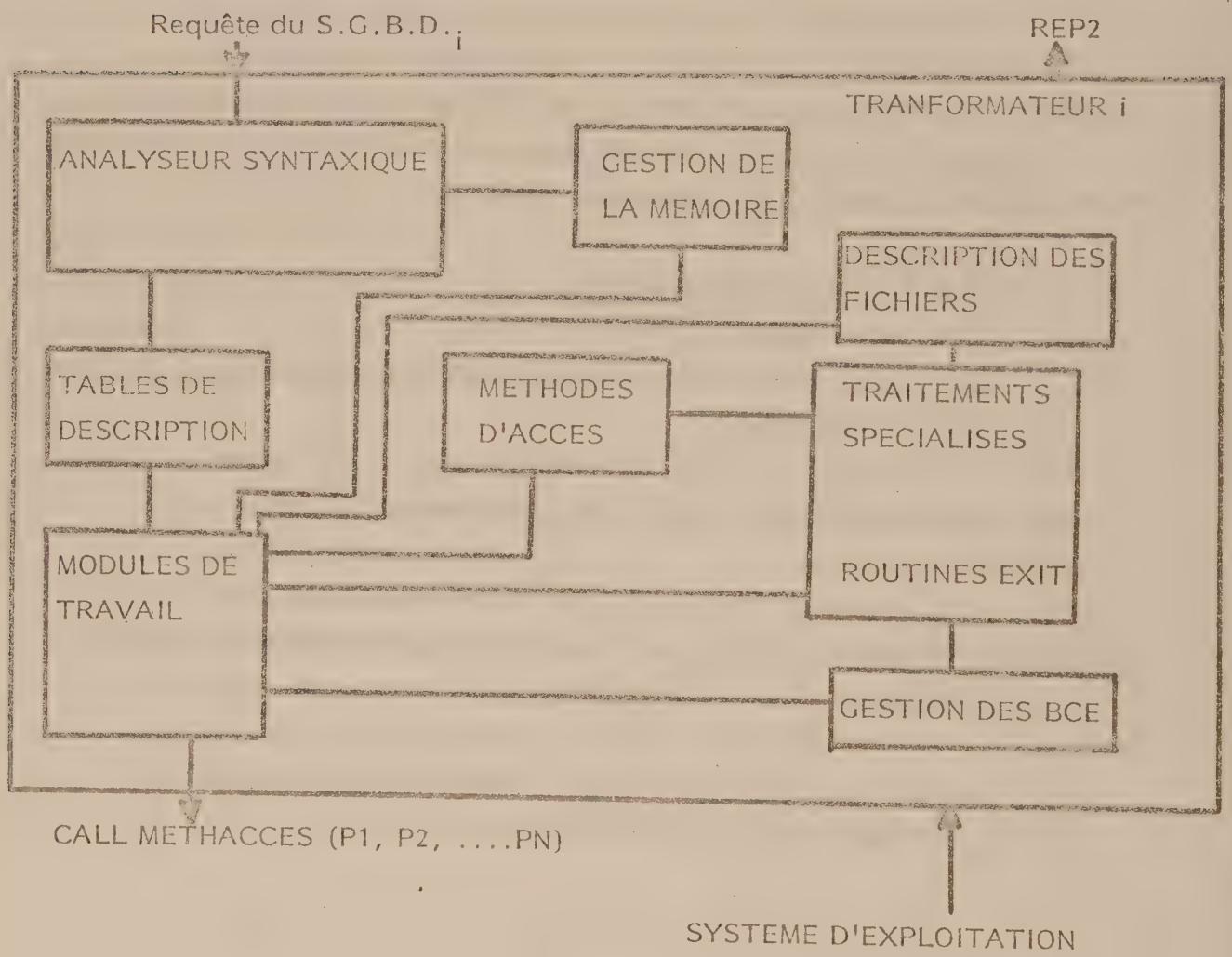



Figure II-6 : architecture d'un transformateur.

Il apparaît clairement qu'un transformateur du S.E.S.G. est constitué essentiellement de:

### II-B-3.1 TABLES :

Ces tables permettent au système :

- de disposer de la description des fichiers ou des bases de données traitées.

- d'avoir pour chaque fichier (ou sous-base) des indications sur les supports physiques et les méthodes d'accès utilisées.

- de gérer les espaces "adresse" des S.G.B.D..

Le format précis de ces tables est donné dans la partie II-C.

### II-B-3.2 MODULES :

Ces modules assurent:

- l'analyse syntaxique et sémantique des requêtes.
- la gestion de la mémoire centrale.
- la gestion des BCE (blocs de contrôle des événements) éventuellement.
- la préparation et le traitement des requêtes émises à destination des systèmes d'exploitation.

## II-C SPECIFICATIONS INTERNES DU S.E.S.G.

### II-C-1 CONCEPT ET DESCRIPTEURS DE MEMOIRE VIRTUELLE.

Dans le paragraphe II-A-3, nous avions défini une méthode d'accès unique pour les S.G.B.D. utilisant notre système.
Un ensemble de tables et de modules du S.E.S.G. permet d'établir la relation entre l'espace des adresses virtuelles et l'espace de la mémoire réelle.
Nous appellerons cet ensemble Machine de Mémoire Virtuelle (MMV).

Le format des tables ou descripteurs de cette machine MMV est indépendant du type d'organisation choisi pour la base de données.

Cette machine utilise essentiellement:

1) des descripteurs de stockage d'information permettant d'accéder aux entités de la base, une entité étant soit un enregistrement de la base de données soit une table de MMV.

2) une zone mémoire centrale gérée dynamiquement grâce à certaines techniques et algorithme que nous verrons.

Il est inconcevable de constituer un descripteur qui contiendrait autant d'entrées qu'il y a de segments dans le système de base de données.

C'est pourquoi, le mécanisme d'adressage de MMV utilise trois types de descripteurs ou tables sur trois niveaux différents.

Au premier niveau, la table des sous-bases (fichiers) permet d'accéder à tous les segments composants une sous-base. C'est une suite de pseudo-registres de base.

Comme dans chaque sous-base, il existe trop d'entités pour qu'il soit envisageable d'attribuer une seule table de segments par sous-base, nous associons à chaque sous-base:

- au deuxième niveau une table des tables de segments.
- et au troisième niveau une ou plusieurs tables de segments.

La structure de chaque table est donnée dans ce paragraphe. La table des sous-bases sera désignée par TSB, une table des tables de segments par TTSG et une table de segments par TSG.

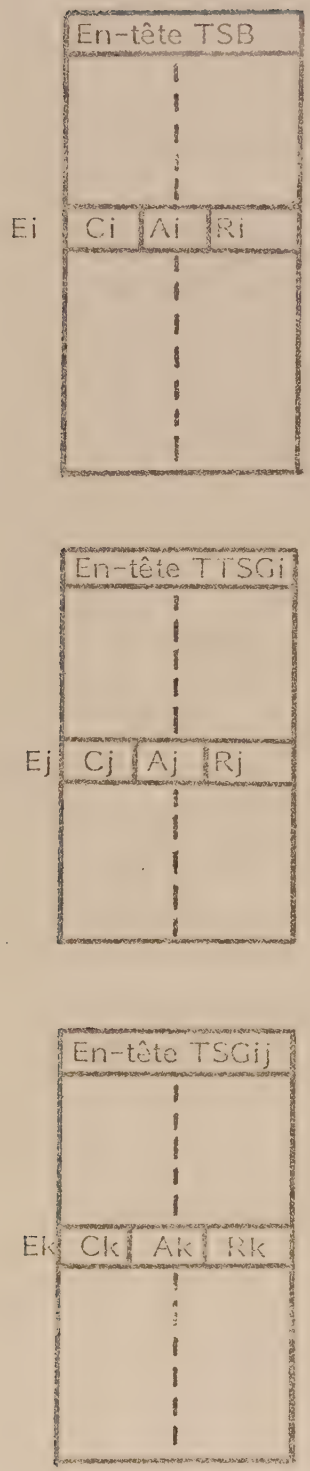



TABLE DES SOUS-BASES

T S B

Ci: zone de contrôle de la TTSGi

Ai: zone-adresse de la TTSGi

Ri: zone de renseignements de la TTSGi

TABLE DES TABLES DES SEGMENTS

T T S G

Cj: zone de contrôle de la TSGij

Aj: zone-adresse de la TSGij

Rj: zone de renseignements de la TSGij

TABLE DES SEGMENTS

T S G

Ck: zone de contrôle de l'entité Sijk

Ak: zone-adresse de l'entité Sijk

Rk: zone de renseignements de l'entité Sijk

Figure II-7: descripteurs de MMV

Ei, Ej et Ek sont appelés descripteurs des tables TSB, TTSG et TSG.
Les entités contiennent des informations spécifiques à chaque table (chaînage des entrées libres, nombre d'entrées ....).

Avant d'étudier avec précision le contenu de ces descripteurs, un schéma du mécanisme d'adressage d'un segment S est donné ci-dessous:

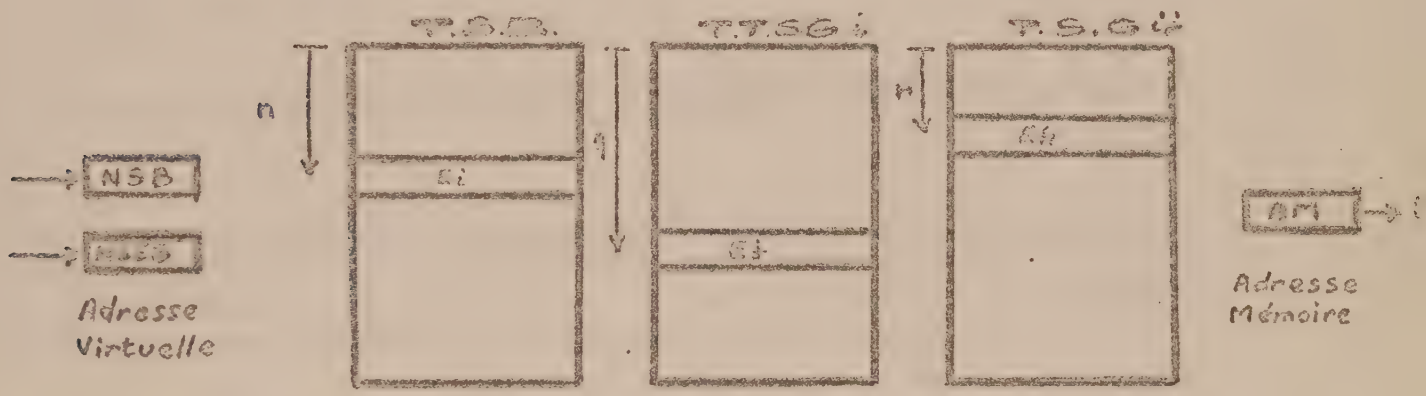



Figure II-8 : mécanisme de traduction

Mécanisme:

Adresse virtuelle $\xrightarrow{f}$ Adresse mémoire. f est la fonction "translation adresse".

Le couple (NSB, NSEG), relatif à un numéro de sous-base (ou nom de fichier) et à un numéro de segment dans la sous-base, représente l'adresse virtuelle.

Les valeurs (n, s) sont placées dans les registres NSB et NSEG.

Si l'entrée n de la TSB est vide, il y a interruption d'absence de TTSG.

$q = \lfloor s/t \rfloor$ t= taille TSG
$r = R_t(s)$

Si l'entrée q de la $TTSG_i$ est vide, il y a interruption d'absence de TSG.

Si l'entrée r de la TSG est vide, il y a interruption d'absence de segments.

AM est chargé dans le registre adresse mémoire.

Remarque :
Les TTSG et les TSG sont des segments du point de vue de MMV. Ces tables appartiennent à la sous-base spéciale réservée à cet effet.

### II-C-1.1 DESCRIPTEURS DE MMV Machine du S.E.S.G.:

Quel que soit le type de la fonction translation-adresse et quelle que soit la technique adoptée pour la réaliser, les modules de MMV ont besoin d'un certain nombre d'informations pour s'exécuter.

Ces informations nécessaires à la bonne gestion de la mémoire virtuelle sont stockées, comme nous l'avons vu précédemment, dans les descripteurs des tables TSB, TTSG, et TSG dans le cas de MMV.

Figure II-9 : Format général d'un descripteur

(a) (b) (c)

a- CONTROLE (1 octet)

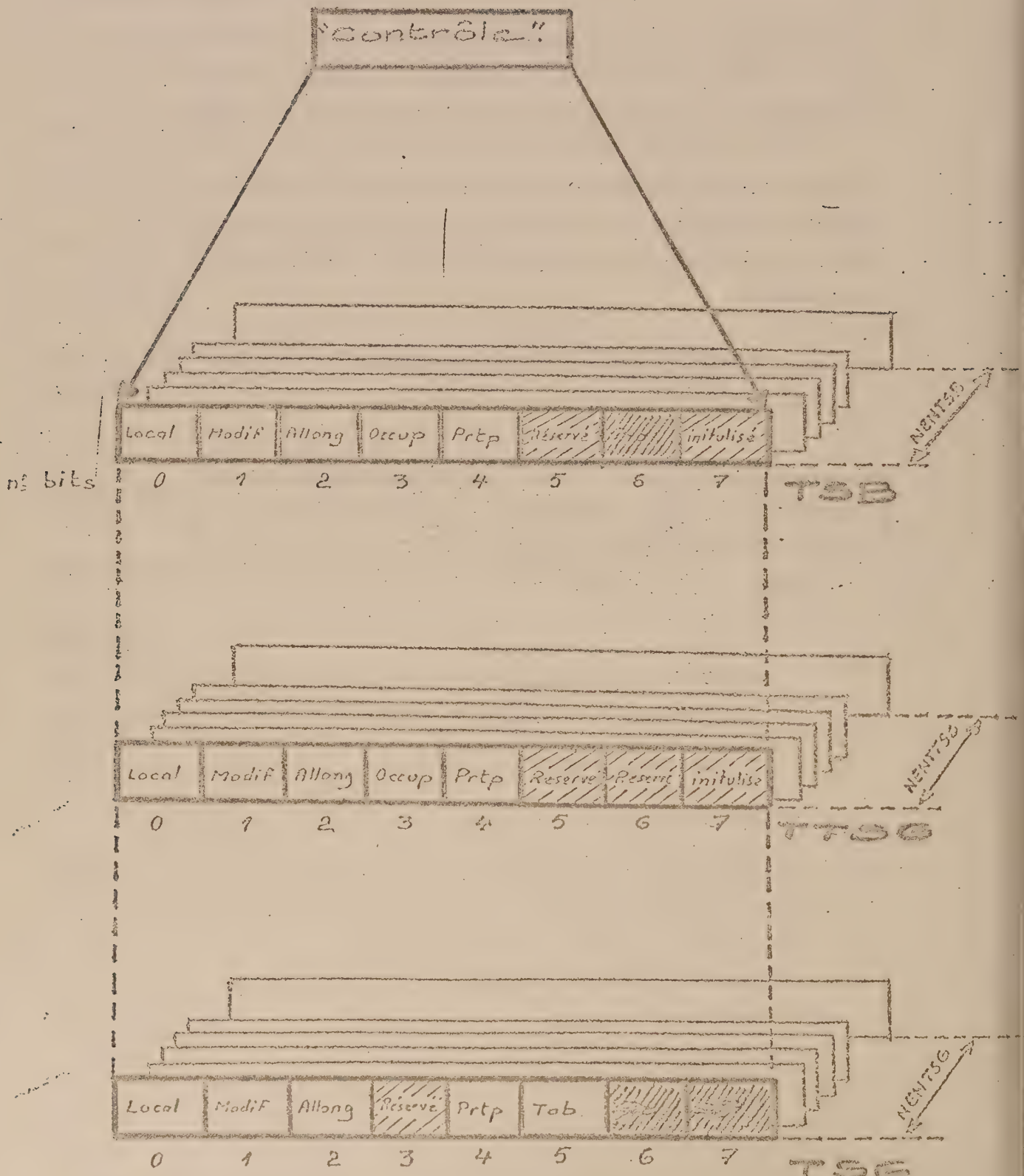



Figure II-10: Décomposition de la zone 'CONTROLE'

Cet octet important permet de contrôler le bloc physique (TTSG, TSG, S) relatif au descriptif qui le contient.

Bit 0

si LOCAL = 0 l'entité physique rattaché (TTSG, TSG, ou S) est absolue en mémoire centrale

si LOCAL = 1 l'entité physique est présente en mémoire centrale

Il n'est pas nécessaire qu'un segment figure autant de fois en mémoire rapide qu'il est demandé. LOCAL nous permet de savoir s'il existe une version de ce segment à une adresse connue en mémoire centrale.

Bit 1

si MODIF = 0 la version de l'entité (TTSG, TSG, S) se trouvant en M.C. est identique à celle qui se trouve sur la mémoire de la masse.

si MODIF = 1 la version de l'entité se trouvant en M.C. est différente de celle de la mémoire en masse.

Bit 2

Ce bit n'a de signification que si et seulement si MODIF vaut 1.

Nous savons qu'il est impossible de réécrire sur M.A.S. (disque par exemple) un enregistrement allongé ou raccourci à l'adresse où il était implanté. (Ceci pour des raisons évidentes de positionnement des GAP ou séparateurs de blocs sur M.A.S.)

si ALLONG = 0 l'entité respective (TTSG, TSG, S) a une taille inchangée. Comme MODIF = 1, il faut exécuter un "REWRITE" à l'ancienne adresse M.A.S. si une mise à jour de cette version est nécessaire.

si ALLONG = 1 l'entité physique a changé de taille. MODIF étant égal à 1, nous ferons un "WRITE" pour toute mise à jour de l'enregistrement qui qui aura une nouvelle adresse M.A.S.

Il est à noter que dans ce cas, l'ancien emplacement de l'entité est inutilisable à priori.

Bit 3

Ce bit est non significatif dans le descripteur relatif à un segment.

si OCCUP = 0
dans le cas du descripteur rattaché à une TSG (entrée d'une TSG): il existe des entrées non occupées dans cette table. Une entrée non occupée étant une entrée dont le contenu est sans signification.

dans le cas du descripteur rattaché à 1 TTSG (entrée de la TSB): il existe des entrées dans cette TTSG permettant d'accéder à 1 ou plusieurs TSG ayant des entrées libres.

si OCCUP = 1
dans le cas du descripteur de la TSB relatif à une TSG, cela signifie que toutes les entrées de cette TTSG "pointent" sur des TSG saturées.

Il est clair qu'avec cette politique, la recherche d'une entrée libre pour insérer une nouvelle entité par exemple devient très rapide.

Bit 4

Ce bit permet aux modules de gestion dynamique de la zone mémoire de M.M.V. de fonctionner avec plus d'efficacité. Nous reviendrons sur ce point.

PRTP = 0

la priorité temporaire d'occupation de place en mémoire est nulle. L'entité relative peut être éventuellement renvoyée sur M.A.S. (d'autres conditions sont à remplir avant le renvoi).

PRTP = 1

l'entité correspondante ne doit pas être renvoyée sur M.A.S. Les modules de récupération de place en mémoire considère que cette entité est de priorité maximale.

Bit 5

Il n'est significatif que dans les descripteurs relatifs aux segments.

TAB = 0

signifie que le segment en question est un enregistrement de la base de données.

TAB = 1

le segment correspondant est une table du système S.E.S.G.

b- "Adresse"

La taille de cette zone dépend essentiellement du type d'adressage adopté. Supposons que cette machine M.M.V. est relative à un transformateur du S.E.S.G. associé à un matériel IBM et que la méthode d'accès du système d'exploitation soit le REGIONAL/3 (n° piste + clé)

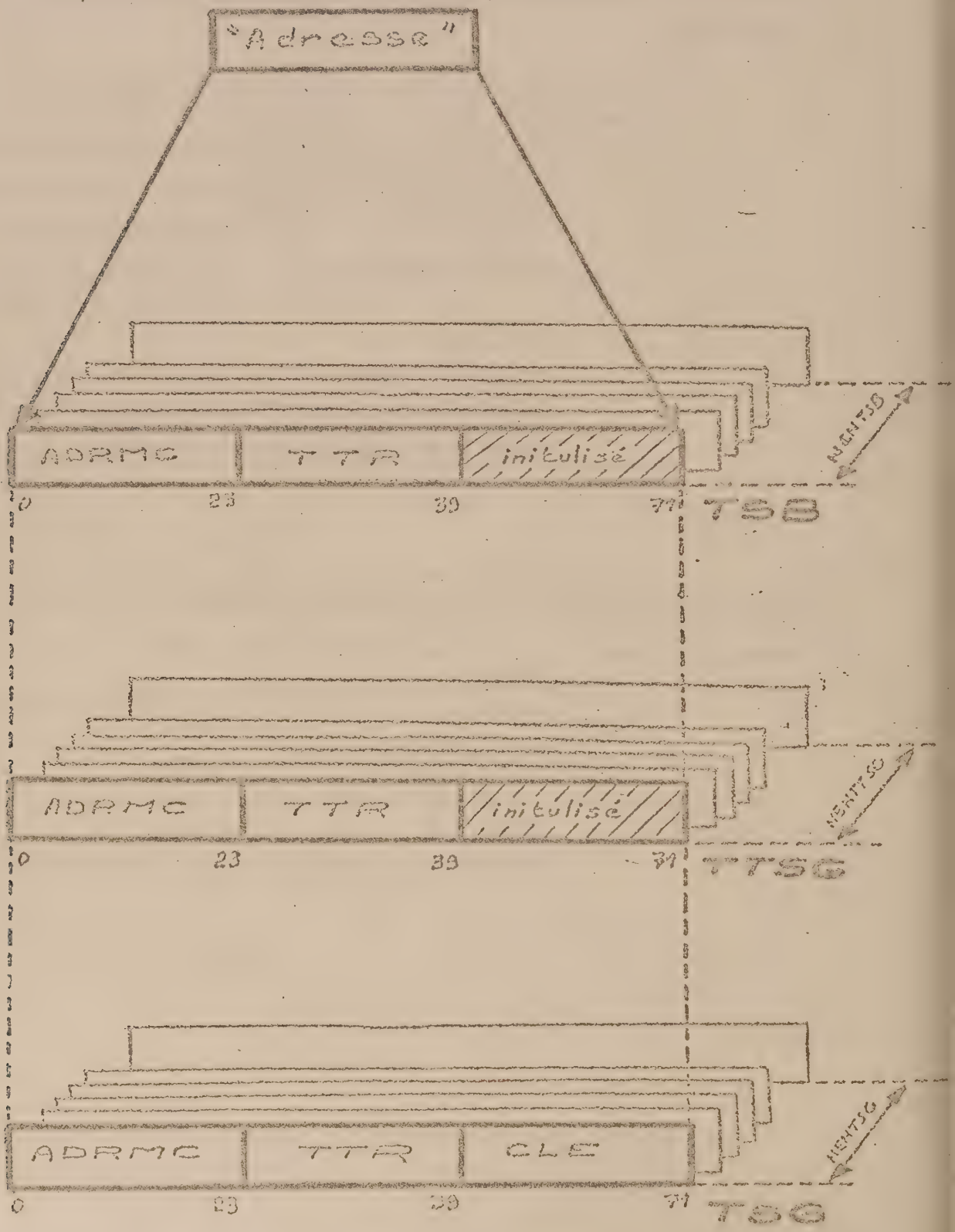



Figure II-11 : décomposition de la zone "ADRESSE"

La zone "ADRMC" est susceptible de recevoir une adresse réelle (mémoire centrale). Elle n'aura de signification que si le bit 0 de la zone "CONTROLE" est positionné à 1.

Dans le cas de M.M.V. se trouvant dans le transformateur du S.E.S.G. précédemment défini, l'encombrement de "ADRMC" est de 3 octets.

Lorsque "LOCAL" est égal à 0, l'entité physique relative au descripteur en question est stockée sur la mémoire de masse. Pour retrouver cette entité, M.M.V. utilise les champs "TTR" et "CLE". Il est clair que ces champs sont spécifiques au type de support secondaire et à la méthode d'accès choisie.

Supposons que M.M.V. stocke les entités physiques sur disques (2314 ou 3330 IBM) et que la méthode d'accès soit la méthode BDAM (Basic Direct Access Method) en option "piste relative" + "clé".

Dans ce cas précis, nous aurons:

Dans la zone "TTR" le numéro relatif de piste par rapport au début du data set, piste sur laquelle se trouve l'entité physique.
Cette zone occupe 2 octets.

En revanche, la zone "CLE" occupe 4 octets. Elle contient la clé effective de lecture/écriture du bloc physique attaché.

Dans le cas de la TSB et de la TTSG, cette zone est inutilisée.
M.M.V. attribue alors une valeur implicite à la clé de lecture/écriture des tables TTSG et TSG.

Prenons l'exemple de la TTSG relative à l'entrée NSB de la TSB:

CLE = 'A' || NSB ;

Alors que pour une TSG attachée à l'entrée NSB de la TSB et NTTSG de la TTSG :

CLE = 'B' || NSB || NTTSG ;

Remarque:

Le couple TTR + CLE est donc unique pour chaque entité physique de la base de données.

Les cas de synonymie sont détectés et traités par la machine M.M.V. par renvoi d'un code de retour au S.G.B.D. émetteur de la requête.

c- "Renseignements"

C'est une zone du descripteur contenant divers renseignements spécifiques au bloc physique attaché et nécessaire au bon fonctionnement du système. Elle peut être schématisée ainsi:

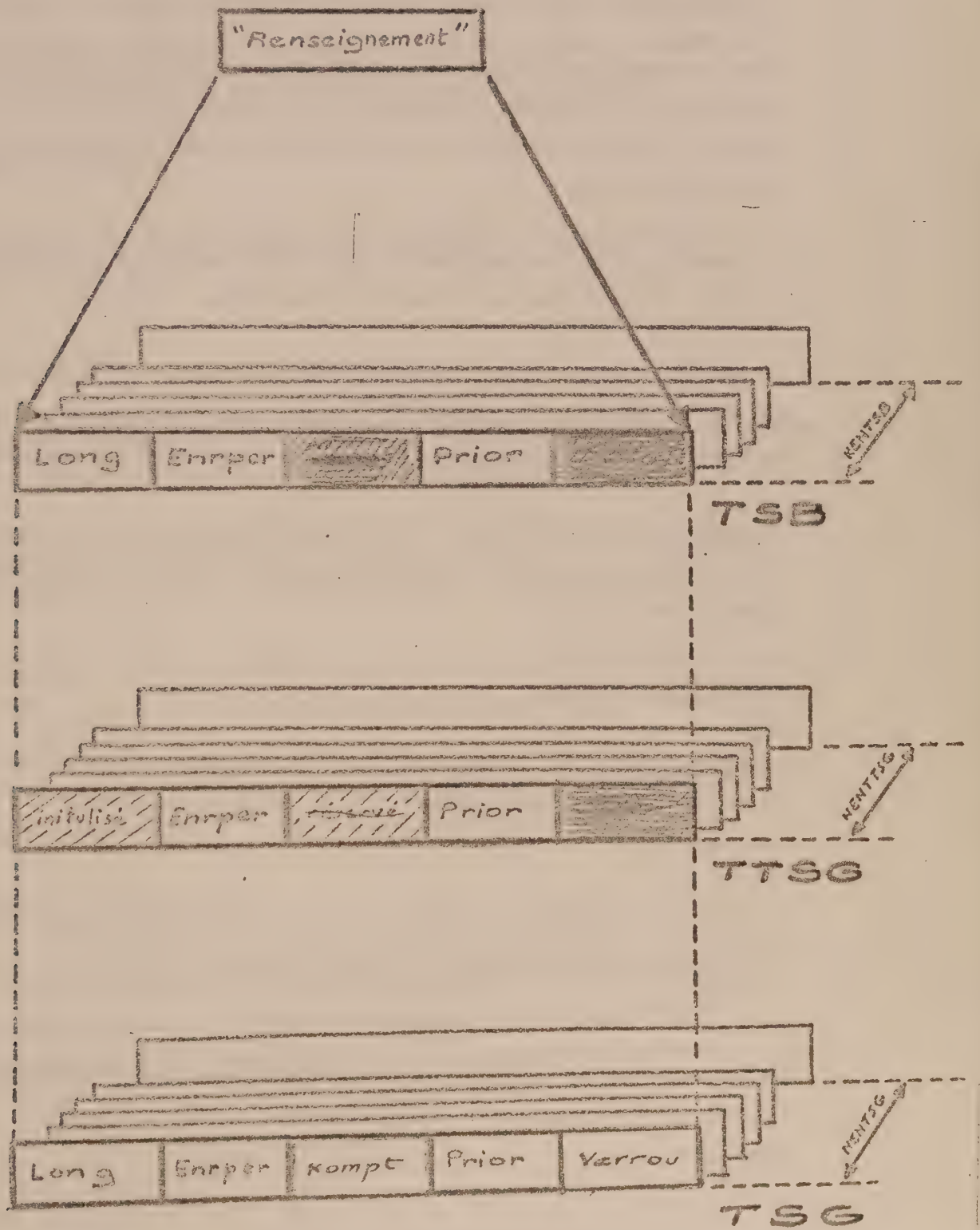



Figure II-12 : décomposition de la zone "RENSEIGNEMENT"

x LONG (2 octets)

Dans le descripteur de la TSB, cette zone contient le nombre d'entrées de la TTSG relative.

Dans le descripteur de la TTSG, cette zone est inutilisée, le nombre d'entrées des TSG étant un paramètre du système S.E.S.G..Il est connu et permet d'exécuter la fonction translation-adresse citée dans les pages précédentes.

En revanche, dans le descripteur de la TSG, cette zone contient la taille en octets du bloc physique attaché à cette entrée.

La valeur explicite (dans TSB et les TSG) ou implicite (dans les TTSG) permet aux modules de gestion de l'espace mémoire centrale d'allouer et de restituer correctement la place dans la zone de travail du système de fichiers.

La gestion de cet espace sera faite essentiellement grâce aux deux routines DEM (allocation) et RESTI (restitution) que nous analyserons dans le paragraphe II-C-1.2.

x ENRPER (2 octets)

Au niveau de la TSB, cette zone jouera le rôle d'un compteur d'enregistrements "perdus" sur le data-set de la sous-base correspondante.

En effet, lors de certaines mises à jour de la base de données, les suppressions d'articles et de tables se feront grâce à la macro instruction DELETE.

Cette macro permet simplement de positionner un "flag" en tête de l'enregistrement sans pour autant le supprimer sur le support physique.

La place est donc perdue.

ENRPER sera incrémenté à chaque fois que ce cas se présentera.

Les S.G.B.D. utilisant le S.E.S.G. doivent pouvoir gérer eux-mêmes ces problèmes de place perdue (au niveau de leur mémoire virtuelle). Cette zone ENRPER servira uniquement en qualité de contrôle.

x KOMPT (2 octets)

Compteur des requêtes de lecture du segment avec "intention de mise à jour" (cf. II-C-2)

x PRIOR (1 octet)

Zone servant à l'établissement et à l'attribution d'une priorité d'occupation de place mémoire centrale gérée par les modules DEM et RESTI.

La libération de place en mémoire centrale se fera alors en fonction de la priorité d'occupation en commençant par la plus basse.

Il est clair que la valeur de $PRIOR_{TSB}$ est strictement supérieure à celle de $PRIOR_{TTSG}$ qui est elle-même strictement supérieure à celle de $PRIOR_{TSG}$.
Cette zone sera consultée par les modules ad-hoc si et seulement si la priorité temporaire PRTP (zone "contrôle") vaut zéro.

x VERROU (1 octet)

Zone du descripteur des TSG relative au problème de verrouillage des enregistrements au moment des mises à jour simultanées par plusieurs S.G.B.D. La technique de gestion est donnée plus loin.

Finalement, un descripteur de TSB, TTSG et TSG fournit toute l'information nécessaire pour la bonne gestion de la mémoire. D'autres tables du S.E.S.G. se greffent sur celles-ci. Elles sont décrites plus loin.

Quant à la longueur totale de chaque descripteur, nous avons:

1 octet pour la zone "contrôle"
9 octets pour la zone "adresse"
8 octets pour la zone "renseignements"

---

soit 18 octets au total pour chaque descripteur.

### II-C-1.2 Mécanismes et Algorithmes de Gestion de l'Espace Mémoire Réelle du S.E.S.G.

Nous allons examiner maintenant les mécanismes utilisés par M.M.V. pour la gestion de l'espace mémoire centrale.

Il convient de signaler que la zone de travail ZDR est de taille fixe pendant tout le temps de fonctionnement du système.

Sa taille est un paramètre du système. Elle est fixée au moment de l'initialisation.

Différentes techniques existent pour l'utilisation de ZDR. Elles se subdivisent en:

- Techniques de remplacement: c'est à dire les techniques de création de nouvelles régions de mémoires à allouer appelées TROUS dans la suite de ce rapport.

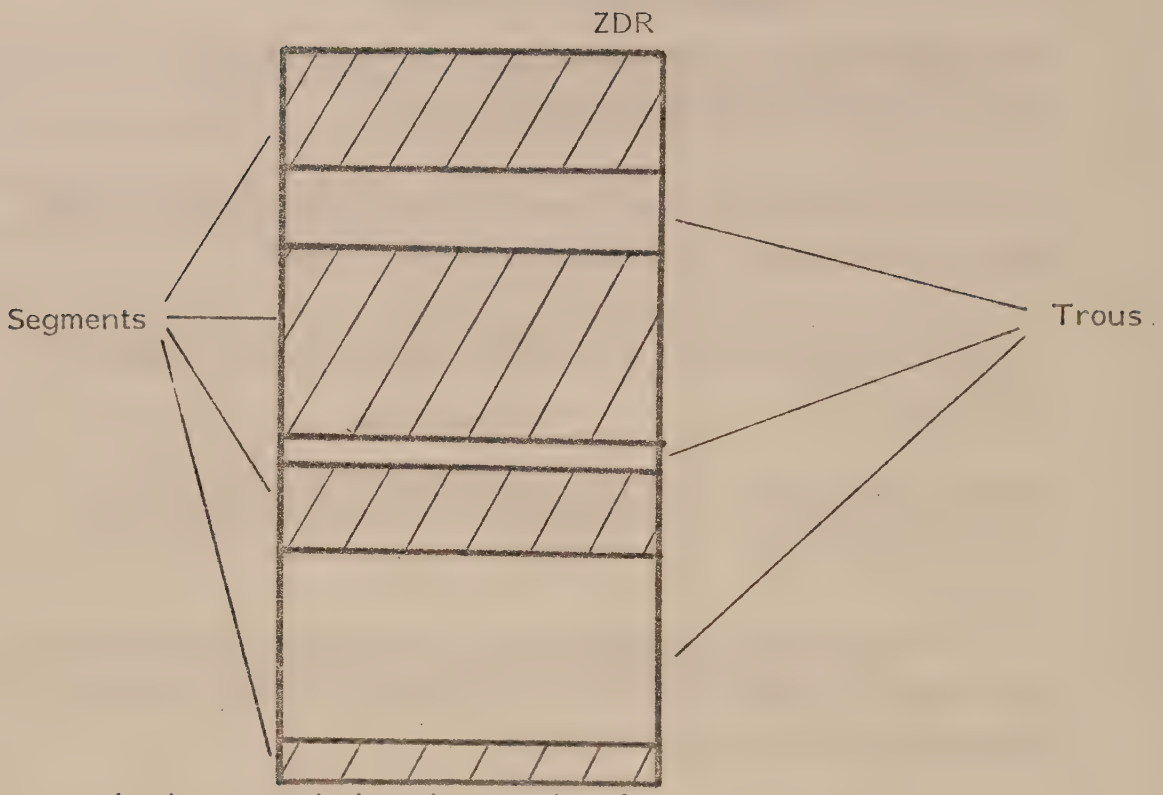
- Techniques de recherche: lorsqu'une demande de place en mémoire est émise.

- Techniques de localisation: concernant les zones où doit être stockée l'information (segments).

Ces techniques sont nombreuses, nous allons citer quelques unes parmi les plus importantes en insistant sur celles adoptées par notre machine M.M.V.

Théorie de 50% due à KNUTH: cette théorie suppose que la mémoire est en équilibre, c'est à dire qu'elle contienne une moyenne de n segments et h trous quand h et n sont grands. Alors h est approximativement égal à n/2.

$$h \simeq n/2$$



zone mémoire centrale à un instant donné.

Figure II-13

Supposons que l'on ait h trous de taille $x_1, x_2, x_3 \ldots\ldots x_h$

a- BEST FIT: la table des trous (ou chaînage) est organisée suivant la taille croissante de ces trous.

b- FIRST FIT: Knuth a constaté que la recherche d'une zone de taille S et l'occupation du premier trou tel que $S \leq x_i$ sans pour cela réorganiser la liste, entraînent après un certain nombre d'opérations une accumulation de petits trous au début de cette liste circulaire.

Le principe du FIRST FIT consiste à avancer le pointeur à chaque opération sur les trous.

Knuth a établi un autre algorithme, c'est celui du:

c- BUDDY SYSTEM: qui consiste à organiser la liste des trous en k listes circulaires contenant chacun des trous de tailles identiques, $2_l$ . $2_r$, ..... $2_k$.

Un trou peut passer de la liste (i+1) à la liste (i) par la création de 2 "buddies" ayant des tailles égales à la moitié de celle du premier trou.

La réciproque est vraie pour transférer deux trous de la liste (i) à la liste (i+1).

Pour trouver un trou de la taille $2_i$, il suffit d'utiliser la procédure récursive suivante:

```
PROCEDURE  : PRENDTROU (i) ;

BEGIN      SI     i=k +1  ALORS RETOUR
       SI   LIST-i = VIDE ALORS
            BEGIN : TROU:= PRENDTROU (i+1);
                        DIVISION DU TROU EN DEUX "BUDDIES" ;
                        MISE DES "BUDDIES" DANS LA LISTE i ;

            END;
       PRENDTROU:= PREMIER TROU DANS LA LISTE i ;

       END;
```

Il est clair qu'un problème de compactage est inévitable et se pose après un certain nombre d'opérations de gestion de l'espace mémoire centrale.

La règle de la mémoire inutilisée nous permet de tolérer une certaine perte de mémoire. De plus, au bout d'un certain temps, les trous sont suffisamment petits pour que certaines insertions soient impossibles. La solution habituellement proposée est le compactage de la mémoire.

Des expériences de simulation ont montré que lorsque la mémoire était pleine, l'algorithme "first fit" détecte la saturation et le compactage n'est pas très utile. En d'autres termes, une bonne stratégie de placement tend à supprimer la nécessité de ce compactage.

Nous verrons un peu plus loin que M.M.V. utilise une stratégie "classique" type FIRST-FIT pour la gestion de l'espace mémoire centrale. Lorsqu'il y a interruption de saturation de cet espace M.M.V. donne le contrôle à une routine qui ne compacte pas la zone mais renvoie sur mémoires secondaires un certain nombre de segments en fonction de règles de priorité décrites lors de l'analyse.

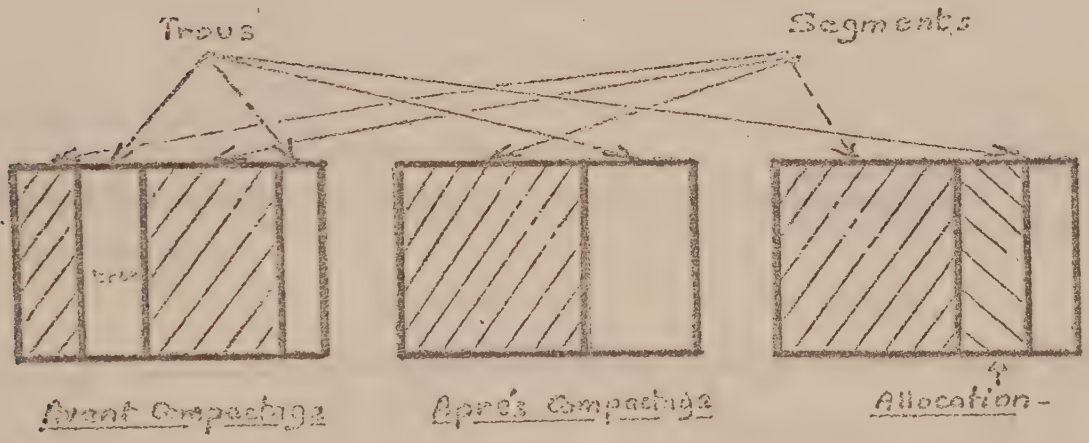
Mais, on peut adopter ultérieurement dans M.M.V. un point de vue inverse, en l'occurence celui du compactage donc qui permet d'éviter de se poser le problème du choix d'un trou.

Cette stratégie se traduira par l'arrêt des activités du système, à certains instants, au profit du compactage des segments vers les petites adresses.

Résultat du compactage: la mémoire du système est en équilibre. Une fraction f de mémoire est occupée par des trous. Si un segment est référencé en moyenne r fois avant d'être effacé, et si $s_o$ est la taille moyenne d'un segment, en appelant F la fraction de temps que le système consacre au compactage, alors:

$$F \geqslant (1 - f) \ / \ \left[ 1-f+ (F/2) \times (r/s_o) \right]$$

En d'autres termes, si on souhaite le moins de temps possible en compactage, il faudra tolérer une perte de mémoire importante.




Evolution de la zone mémoire centrale
Figure II-14

Plusieurs facteurs entrent en jeu dans le choix de la taille des "pages". Par similitude, le problème est le même dans le cadre des systèmes de bases de données pour le choix de la taille optimum de la zone mémoire centrale (ZDR).

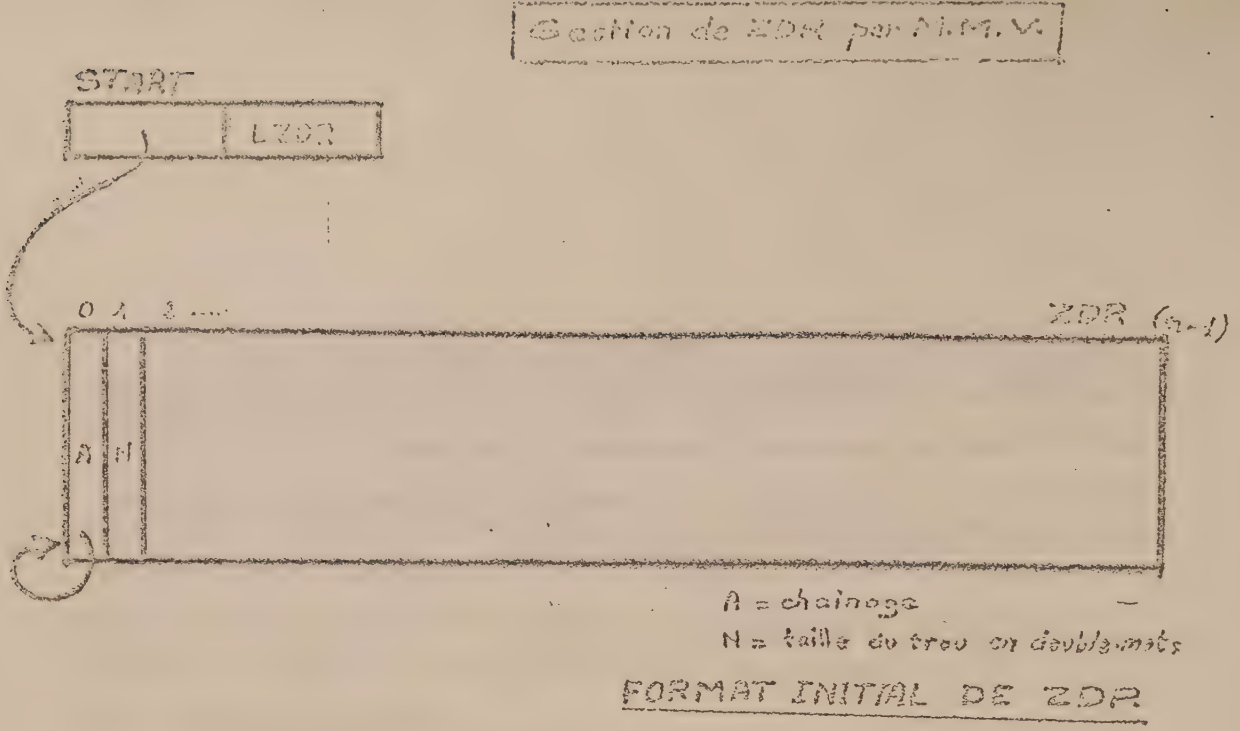



Figure II-15

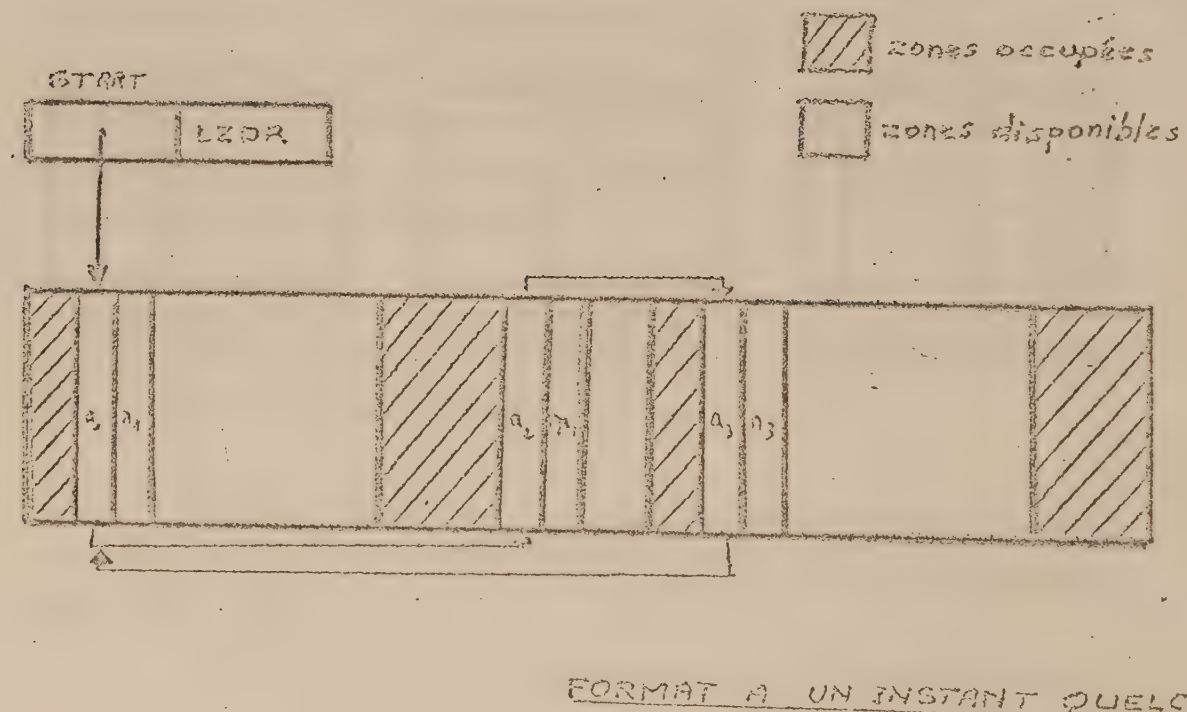



Figure II-16

START représente l'entrée de la liste circulaire des trous.

En fait, la version opérationnelle de M.M.V. ne possède pas de liste circulaire.

Le dernier trou de la liste est |à -1.

Lorsque le système de bases de données s'exécute, toute référence à un segment sera faite à travers l'interface M.M.V.

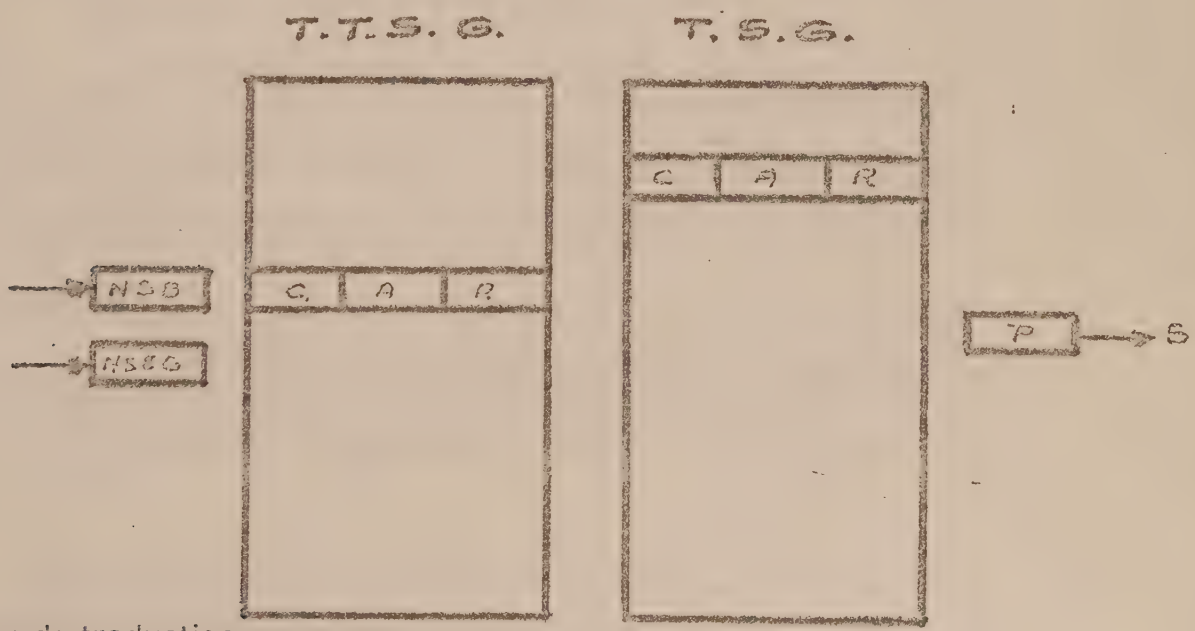
Cette référence permet d'établir la liaison

Adresse Réelle ⟷ Adresse Virtuelle

ou

P ⟷ (NSB, NSEG)

P étant un pointeur en mémoire centrale
(NSB, NSEG) étant un couple qui représente l'adresse virtuelle du segment.




Tables de traduction

Figure II-17

### II-C-1.2.1 Lecture d'un segment

L'appel du S.E.S.G. à partir d'un S.G.B.D. se fait par l'instruction :

```
CALL SESG (NSGBD, NBASE, FIC, 'L', IDTR, PROX, ORGLO, ZES2, REP2, BCE);
```

En échange de l'adresse virtuelle (FIC, IDTR) en fait (numéro de sous-base, numéro relatif du segment), notre système permet de fournir au S.G.B.D. appelant l'enregistrement dans la zone ZES2.

Nous verrons en détail comment le S.E.S.G. procède pour accèder physiquement à l'information.
Cependant, deux cas peuvent se produire.

#### II-C-1.2.1.1 Segment en mémoire:

Après certaines vérifications et contrôles à travers les descripteurs relatifs (segment non verrouillé, (LOCAL) = '1'B, (MODIF) = 'O'B ...) le S.E.S.G. renvoie le segment dans la zone ZES2 à partir de ses zones tampons.

#### II-C-1.2.1.2 Segment manquant:

Dans ce cas précis, ((LOCAL)= 'O'B ou bien (MODIF) = '1' B...), le S.E.S.G. doit effectuer une lecture du segment référencé par (FIC, IDTR).

Pour cela, le S.E.S.G. doit allouer de la place et doit mettre à jour le chaînage des trous de la zone mémoire centrale ZDR.

Ce travail sera effectué par une routine à laquelle le système doit fournir la longueur en octets de l'espace à allouer ( champ "LONG" du descripteur associé) .

L'instruction:

CALL DEM (L, P, CR) ;

permet de chercher un trou $x_i$ tel que L $x_i$ et de nous renvoyer son adresse dans le pointeur P après mise à jour du chaînage des blocs disponibles. CR est un code de retour.

Lorsque DEM ne trouve pas de bloc de taille suffisante, il fait appel à la procèdure LIBERE par l'instruction:

CALL LIBERE (INDIC, CR) ;

Cette routine renvoie sur mémoire secondaire toute entité ayant:

(A) ((PRIOR) $\prec$ (INDIC) avec (PRTP) = '0'B

Ceci suppose que le S.E.S.G. ait pris la précaution de positionner le bit PRTP de toutes les entités en traitements à '1' B.

Lorsque LIBERE redonne le contrôle à DEM plusieurs cas peuvent se présenter:

1- LIBERE a effectivement renvoyé sur mémoire de masse certains blocs et la demande de place peut être satisfaite. DEM continue normalement.

2- LIBERE a effectivement renvoyé sur mémoire de masse certains blocs mais la demande de place ne peut pas être satisfaite alors

DEM réitère son appel à LIBERE autant de fois qu'il sera nécessaire avec:

(INDIC) = (INDIC) + 1 ; ceci jusqu'à (INDIC) = 6

Les tables et segments ayant une priorité 6 seront résidents pendant toute l'exécution.

3- LIBERE n'a pas trouvé de blocs ayant les caractèristiques définies par les équations (A), alors DEM se trouve dans la même situation que précédemment.

II-C-1.2.2 Ecriture d'un segment:

L'ordre d'écriture d'un segment se fait par:

CALL SESG (NSGBD, NBASE, FIC, 'E', IDTR, PROX, ORGLO, ZES2, REP2, BCE);

Sa fonction est inverse de l'ordre précédent.
Cette fonction permet d'écrire sur mémoire de masse l'enregistrement se trouvant à l'adresse mémoire centrale ZES2.

De plus, si (PROX) = 0 le S.E.S.G. doit rechercher un emplacement physique le plus proche possible de l'enregistrement se trouvant à l'adresse référencée par (FIC, PROX).

La technique de recherche d'emplacement est appelée technique de "l'essui -glaces" car S.E.S.G. recherche une place de préférence dans le même bloc physique, puis dans le bloc avant, le bloc arrière, la piste avant, la piste arrière etc.....

Enfin, S.E.S.G. affecte un descripteur à cette nouvelle entité et renvoie son adresse relative (FIC, IDTR).

Quant à la restitution de la place mémoire, elle se fait par:

CALL RESTI (L, P, CR);

Ce module crée dans la zone ZDR un bloc disponible à l'adresse P et ayant une longueur L.

Il chaîne ainsi ce nouveau bloc avec les autres trous existants. Dans certains cas, il fusionne avec ce bloc un ou deux autres s'ils sont adjacents.

Nous trouverons dans la troisième partie des algorithmes de ces modules qui constituent les principaux programmes de la machine M.M.V.

II-C-2 VERROUILLAGE DES SEGMENTS:

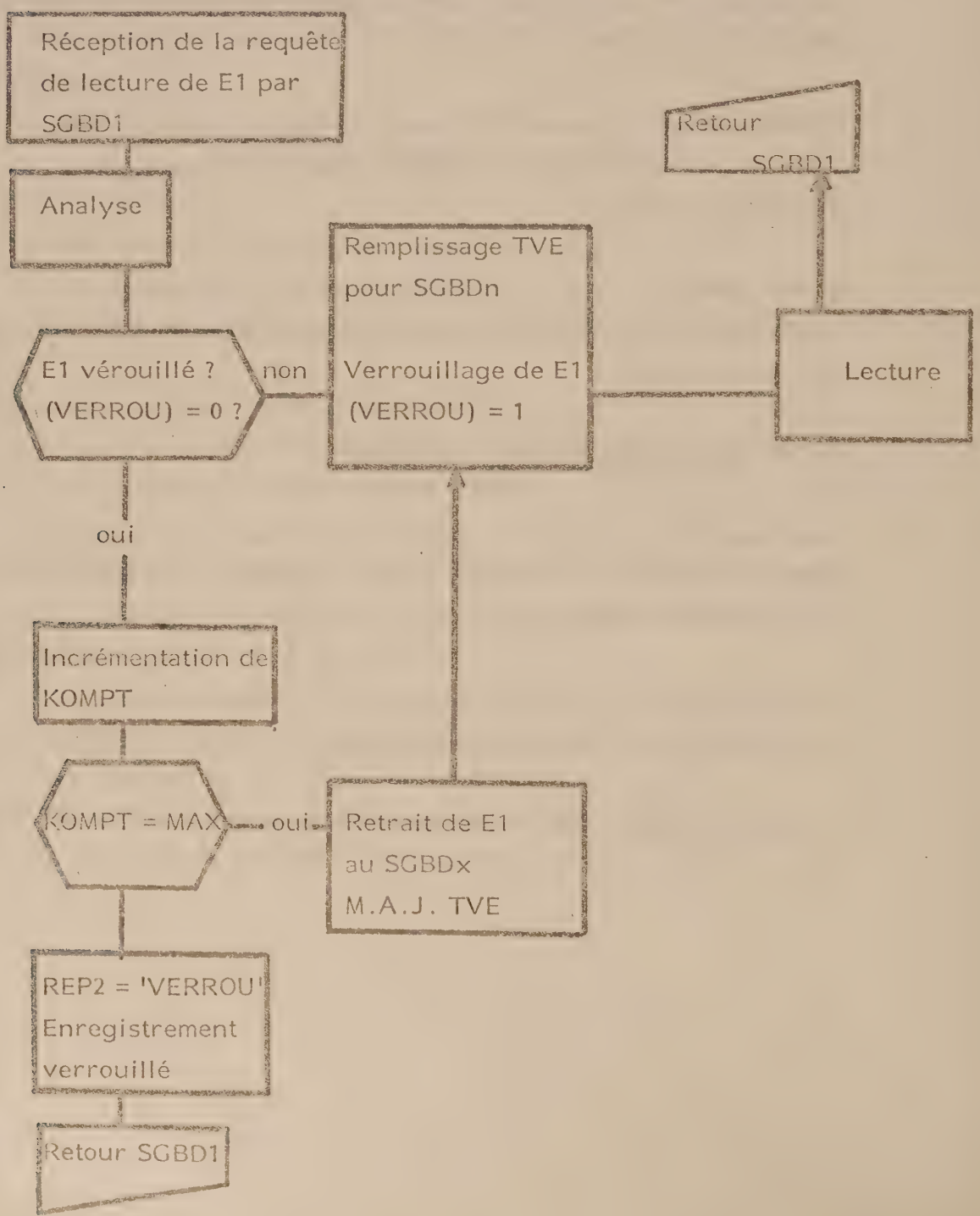
Etant donné que le S.E.S.G. permet l'exécution simultanée de plusieurs S.G.B.D. sur la même base, le problème de protection des segments se pose.

Il a été résolu d'une manière simple par l'utilisation de la zone 'VERROU' du descripteur relatif au segment lu ou en lecture.

En effet, tout appel à S.E.S.G. avec la fonction logique 'L' se fera de deux manières suivantes:

1- le S.G.B.D. lance l'appel de lecture avec "intention de modification" du segment (REP2) = 'INTMAJ'.

Dans ce cas précis, le S.E.S.G. exécute la séquence relative à l'organigramme suivant:



Le S.G.B.D.x, à qui le S.E.S.G. a retiré d'une manière "dictatoriale" l'enregistrement E1, recevra une réponse REP2 = 'RECOMM' après la tentative d'écriture de cet enregistrement qui signifie au S.G.B.D.x de recommencer la lecture.

2- Le S.G.B.D. lance l'appel de lecture sans "intention de modification" du segment ((REP2) = '  ').

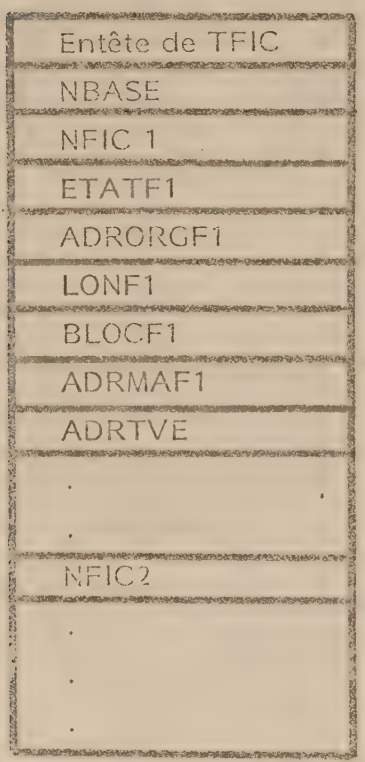
Si cet enregistrement tente tout de même de modifier cet enregistrement, le S.E.S.G. lui enverra un code de retour (REP2) = 'PAINMA' après la tentative d'écriture.

## II-C-3 TABLES DES CONTEXTES:

Des tables de contexte sont nécessaires pour le bon fonctionnement du S.E.S.G.
Dans un même transformateur, plusieurs types de tables sont présentes.

### II-C-3.1 TABLES DES FICHIERS:

Cette table contient pour une base donnée des informations et des adresses de tables d'informations sur tous les fichiers (ou sous-bases) qui la constituent. Cette table s'appelle TFIC.




nom de la base de données
nom du premier fichier de la base
compteur ouverture/fermeture
adresse de la table de l'organisation du fichier
longueur de l'enregistrement logique
facteur de blocage
adresse des méthodes d'accès du fichier
adresse de la table des verrous relative à F1

## II-C-3.2 TABLES DES ORGANISATIONS DE FICHIERS :

Une des caractéristiques du S.E.S.G. est de rendre les systèmes de gestion de bases de données indépendants des supports physiques des fichiers et des méthodes d'accès des constructeurs.

A tout instant, le S.E.S.G. doit pouvoir associer à une adresse virtuelle donnée, une adresse réelle ainsi que le support physique et la méthode d'accès utilisés.

Les tables des organisations des fichiers (TOF) permettent d'établir cette relation.

A chaque fichier ou sous-base est associé une TOF ayant un format du type suivant

| | |
|---|---|
| Entête de la TOF | TOF du fichier 1 |
| 0 I1 | adresses virtuelles de 0 à I1 (domaine 1) |
| DDN1TFOC | nom du "DATA SET" relatif au domaine 1 |
| NUMMAC1 | numéro de l'entrée dans TMA pour le domaine 1 |
| TYPSUP1 | type de support physique relatif au domaine 1 |
| CURSEUR DDN1 | curseur de positionnement dans DDN1 (bande, disque, carte magnétique) |
| | |
| $I_{-1}$ + 1 I2 | adresses virtuelles de $I_1 + 1$ à $I_2$ (domaine 2) |
| DDN2TFOC | nom du "DATA SET" relatif au domaine 2 |
| NUMMAC2 | numéro de l'entrée dans TMA pour le domaine 2 |
| TYPSUP2 | type du support physique relatif au domaine 2 |
| . | |
| . | |

Grâce à cette table, le S.E.S.G. peut distribuer les enregistrements d'un fichier sur un ensemble de supports physiques quelconques suivant des organisations quelconques.

Exemple:

Espace Virtuel de la sous-base F1

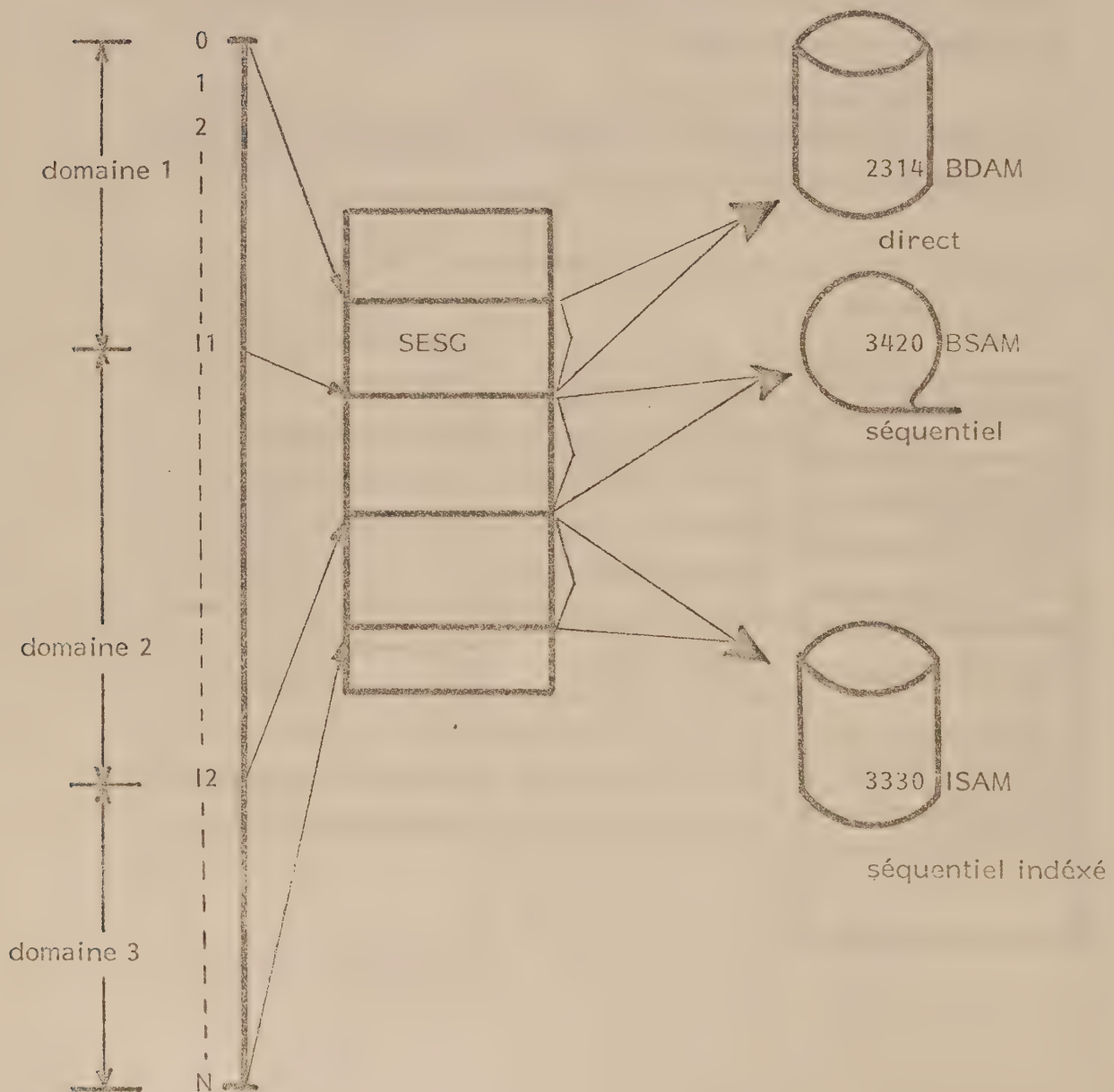



Figure II-18 : indépendance des supports et méthodes d'accès

## II-C-3.3 TABLE DES METHODES D'ACCES :

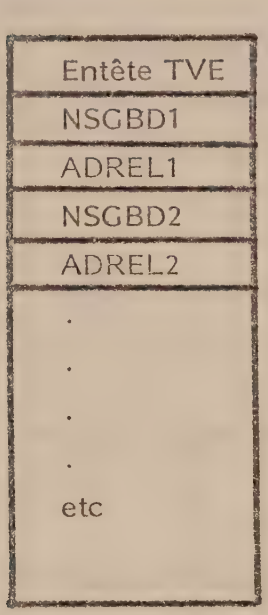
C'est une table associée à un système d'exploitation donné. Cette table est évolutive (TMA) par la création de nouvelles entrées éventuellement.

L'apparition d'une nouvelle méthode d'accès n'entraîne que l'initialisation de nouvelles entrées dans cette table.
Toutes les tables TFIC et TFOC "pointent" sur cette table.

| | |
|---|---|
| Entête de la TMA | |
| NOMMA1 | nom de la méthode d'accès n°1 |
| NUMODULE1 | numéro du module d'accès utilisé |
| NOMMA2 | numéro de la méthode d'accès n° 2 |
| NUMMODULE2 | numéro du module d'accès utilisé |
| . | |
| . | |
| . | |
| . | |

II-C-3.4 TABLE DES VERROUS :

C'est une table relative au problème de protection des segments (cf II-C-2).
Le nombre d'entrées de chaque table TVE est variable. Le format est donné ci-dessous:




Dans un premier temps, nous nous sommes limités à la possibilité de verrouiller qu'un seul segment par S.G.B.D. et par fichier.

Toutefois, plusieurs S.G.B.D. peuvent verrouiller des enregistrements différents dans un même fichier ( 1 enregistrement pour chaque S.G.B.D.).

Le nombre maximum d'entrées d'une TVE est alors égal au nombre total de S.G.B.D. qui s'exécutent simultanément avec le S.E.S.G.

Si un S.G.B.D. lance une lecture d'un segment E1 avec "intention de modification" et qu'il ne récrit pas cet enregistrement la zone ADREL correspondante est écrasée par l'adresse relative de l'enregistrement suivant lu dans les mêmes conditions.

## II-C-4 LIAISONS ENTRE LES TABLES DU S.E.S.G.

Toutes les zones "adresse" des tables sont des zones du type des descripteurs décrit dans la partie II-C-1.1.
La figure suivante indique toutes les liaisons qui peuvent exister entre les tables:

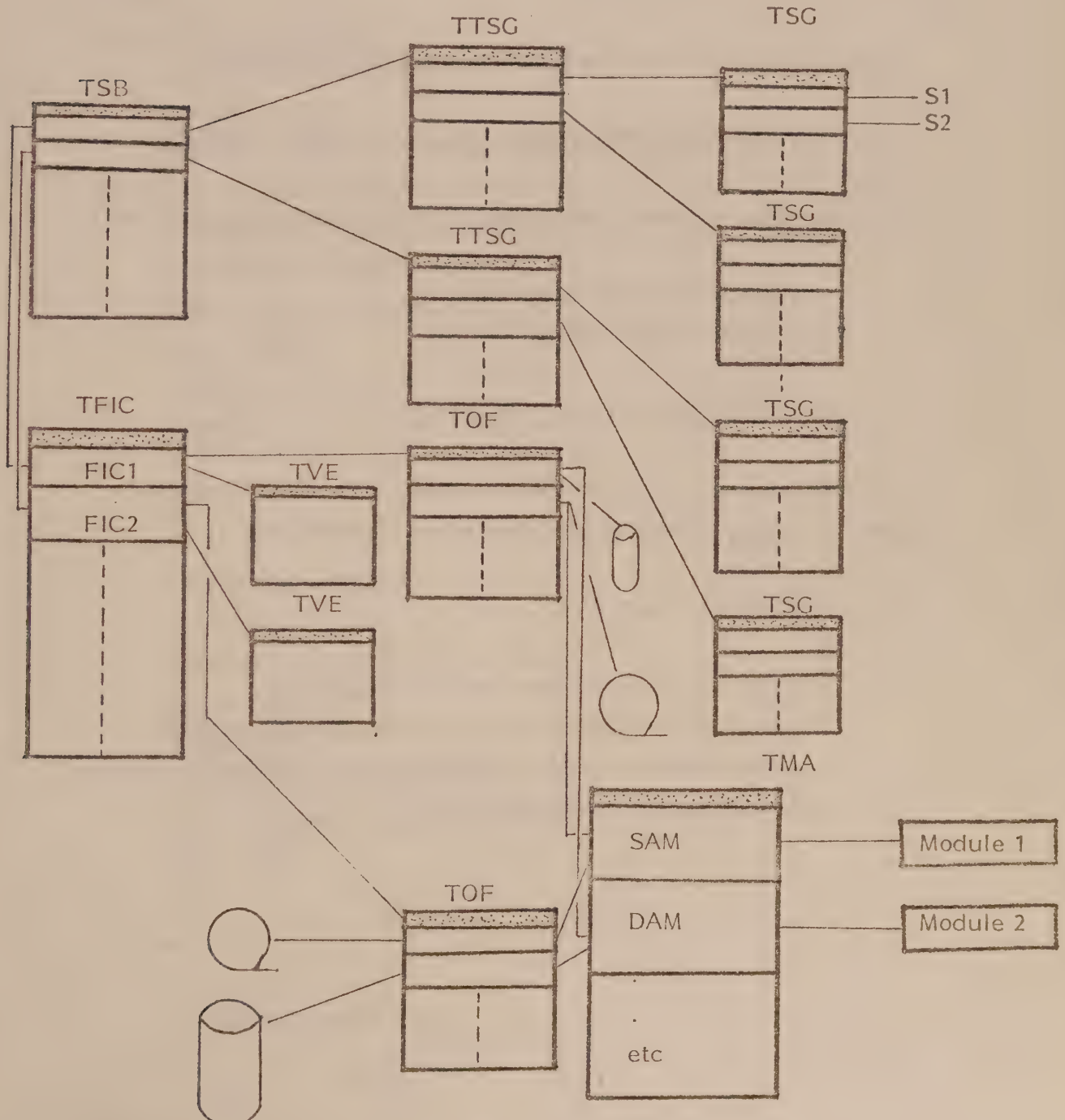



Figure II-19 : liaisons entre tables du S.E.S.G.

Pour chaque base il existera un ensemble de tables de ce type par transformateur.

Il y a une relation biunivoque entre les entrées de chaque TSB et de la TFIC correspondante.

Toutes ces tables ne sont pas créées dans la même étape.

### II-C-5 ETAPES DE GENERATION DES TABLES ET MODULES

Nous distinguons deux grandes phases de création de ces tables:

- la phase "génération système"
- la phase "exécution du S.E.S.G."

#### II-C-5.1 GENERATION "SYSTEME" :

La première phase est relative à la création d'un transformateur associé à un système d'exploitation et un matériel donné.

Il s'agit d'indiquer au S.E.S.G.:

- le nombre de bases de données à supporter.
- le nombre de fichiers maximum par bases de données.
- les partitions de chaque fichier sur les supports physiques.
- les méthodes d'accès utilisables.

Le programme de "génération système" est alors activé:

CALL GENSYS (NSB, NMA); avec

NSB = nombre maximum de sous-bases.
NMA= nombre maximum (à la création) de méthodes d'accès.

Les données en entrée se présentent sous la forme suivante:

```
NOMBASE = XXXXXX
TOTAL-ENR-LOGIQUES = N
        x NOMFICH = YYYYYY
        LIMITES = I1, I2
        SUPPORT ={2314 ou 3330 ou 3420 ou ....}
        MET-ACCES={BSAM ou BDAM ou ....}
        MOD-ACCES ={NOM1 ou NOM2 }
        LONG-ENR-LOGIQUE = n1
        NBRE-ENR-PAR-BLOC = n2
        DDN = XXXX
        LIMITES = I3, I4
        SUPPORT
         ⋮
        etc

        x NOMFICH = ZZZZZZ
        LIMITES = m1, m2
        SUPPORT = {2314, 3330, ou 3420}
        MET-ACCES = ····
        MOD-ACCES =····:
        LONG-ENR-LOGIQUE = n3
        NBRE-ENR-PAR-BLOC = x4
        DDN = XXXX
         ⋮
        etc

        x NOMFICH = XYXYXY
                              SORTIE
                    SYSOUT
```

Tous ces ordres doivent commencer en colonne 2, les caractères blancs ne sont pas autorisés.
Des valeurs par défaut sont attribuées à certains éléments par les modules de GENSYS. Par exemple, l'absence de support relatif au fichier YYYYYY entraîne l'affectation d'un espace disque 2314 à cette sous-base.

GENSYS génère et initialise en fonction de ces données les tables TFIC, TOF, TMA, TSB, et TTSG.

Ces tables sont stockées sur un 'DATA SET' approprié qui sera transmis au S.E.S.G. pour la phase d'exécution.

A chaque méthode d'accès correspond un module qui doit être écrit et stocké dans la librairie du système S.E.S.G.

### II-C-5.2 PHASE EXECUTION:

Pendant l'exécution normale du S.E.S.G., le système crée les tables TSG et TVE chaque fois que cela est nécessaire par appel des modules adéquats.

Les organigrammes donnés dans la troisième partie montre bien le processus adopté pour cette création.

# III- TROISIEME PARTIE

## III-A GENERALITES : LES PROGRAMMES DU S.E.S.G.

## III-B PRESENTATION GENERALE DES PROGRAMMES

## III-C ORGANIGRAMMES ET LISTINGS DES PROGRAMMES

## III-D EXEMPLES D'INTERROGATION DE BASES DE DONNEES

## III-A GENERALITES : LES PROGRAMMES DU S.E.S.G.

Le choix du langage de programmation des modules du S.E.S.G. a posé un problème difficile à résoudre.

En effet, notre système doit satisfaire à plusieurs exigences, tant au point de vue de la transportabilité que de celui des performances d'accès aux données.

Nous avons opté pour le PL/1 car c'est un langage qui a semblé le mieux adapté à notre problème par les facilités qu'il offre au niveau de l'allocation dynamique des tableaux de même au niveau de l'utilisation des chaînes de bits pour les descripteurs de tables.

De plus, ce langage permet des modifications faciles et une clarté dans la compréhension du système.

Nous avons limité notre exemple à un seul transformateur associé à la machine utilisée (IBM 370/158 cf I-B-5.2) pour des raisons pratiques de test.

La version '001' du S.E.S.G. présentée est une version "mono-S.G.B.D."; les zones BCE (blocs de contrôle des événements) sont prévues mais non utilisées.

D'autre part, plusieurs options restent possibles par ajout de nouveaux modules (traitements spécialisés par exemple).

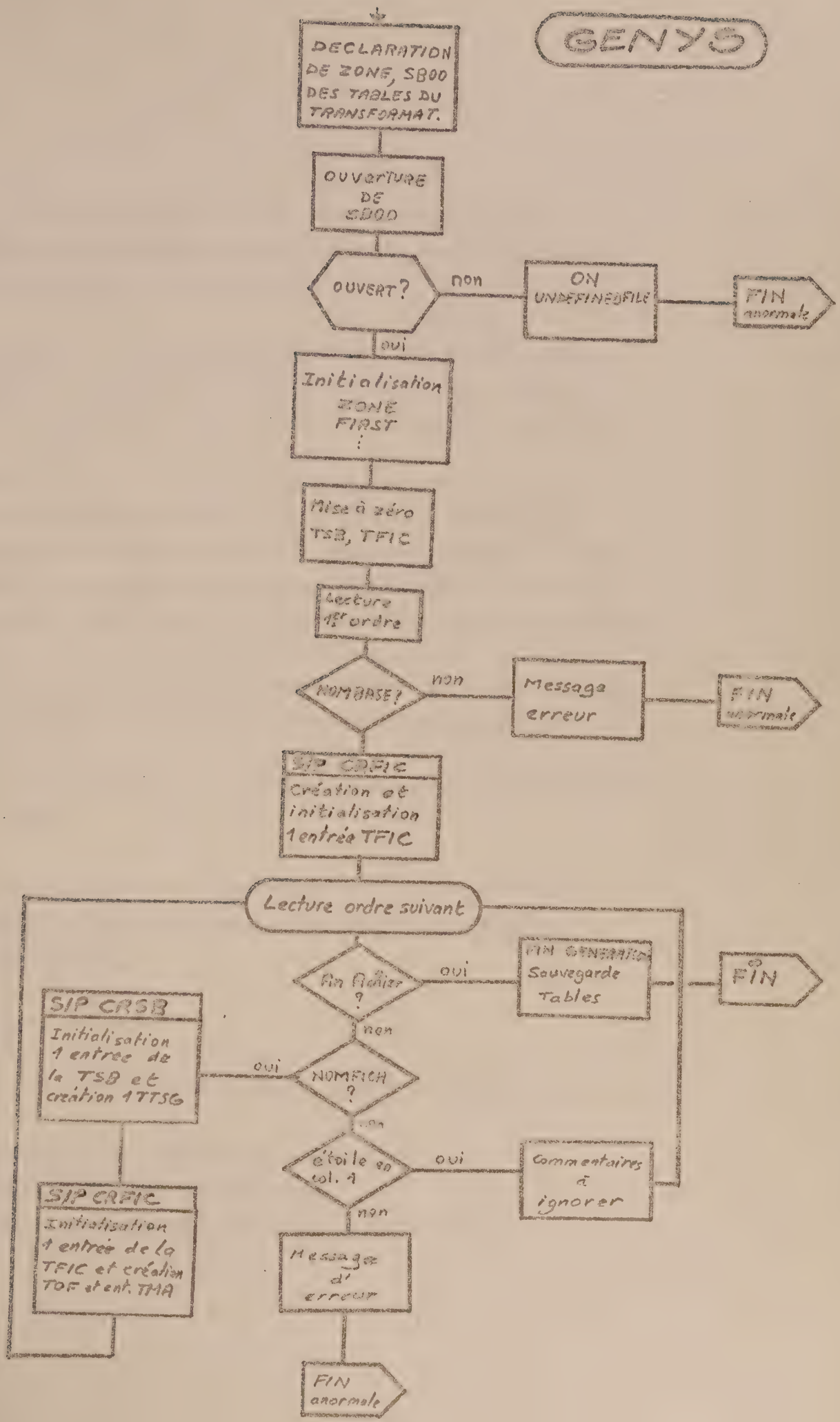
## III-B PRESENTATION GENERALE DES PROGRAMMES :

1- Le programme principal, relatif à la phase "génération système" d'un transformateur donné, lit les données et paramètres de base, prépare les tableaux de descripteurs. Il sauvegarde les "racines" TSB et TFIC dans un 'DATA SET' spécial "SB00" réservé essentiellement aux tables du système S.E.S.G.
Ce programme GENSYS initialise la zone mémoire centrale ZONE.

Il fait appel au sous-programme CRSB chaque fois qu'il lit un ordre "NOMFICH = XXXXXX" qui annonce la création d'une nouvelle sous-base.
CRSB crée une entrée dans la TSB et l'affecte à cette sous-base, il initialise une TTSG puis la stocke dans le fichier SB00.

Après cette phase d'initialisation des tables de gestion de l'espace mémoire GENSYS appelle le sous-programme CRFIC qui crée et initialise les tables TFIC, TOF, et TMA relatives à la nouvelle base de données.

Dans le cas d'une erreur de syntaxe, GENSYS imprime un message d'erreur et abandonne le traitement.


GEN75
DECLARATION DE ZONE, SBOO DES TABLES DU TRANSFORMAT.
OUVERTURE DE SDOO
OUVERT?
non
ON UNDEFINEDFILE
FIN anormale
oui
Initialisation ZONE FIRST
Mise à zéro TSB, TFIC
Lecture 1er ordre
NOMBASE?
non
Message erreur
FIN anormale
SIP CRFIC
Création et initialisation 1 entrée TFIC
Lecture ordre suivant
Fin Réalisation ?
oui
FIN GENERATION Sauvegarde Tables
FIN
non
SIP CRSB
Initialisation 1 entrée de la TSB et création 1 TTSG
oui
NOMFICH ?
non
étoile en col. 1
oui
Commentaires à ignorer
non
SIP CRFIC
Initialisation 1 entrée de la TFIC et création TOF et ent. THA
Message d'erreur
FIN anormale

2- Le programme central (S.E.S.G.) relatif à la "phase exécution du système" est activé comme un sous-programme à chaque requête d'un S.G.B.D.

La version actuelle est non-réentrante. Elle est opérationnelle en mono-S.G.B.D. et consiste à analyser les paramètres transmis et à préparer un appel au sous-programme DISTRI.

Avant de lancer un appel à ce module, le S.E.S.G. définit le type de fonction à exécuter (L, E, S, R), recherche dans les tables du système l'entrée de la TSG correspondante à l'opération (ETSG) par appel du sous-programme CREER qui recherche éventuellement une entrée libre (cas d'une écriture par exemple).

Le S.E.S.G. recherche également à partir de la table des fichiers (TFIC) l'entrée (ETOF) de la table des organisations du fichier concerné.
Le S.E.S.G. détermine l'entrée de la table des méthodes d'accès à utiliser.

Toutes ces adresses et paramètres sont transmis au module DISTRI qui est le seul programme capable de lancer les modules d'E/S.

L'appel se fait par :

CALL DISTRI ( OPE, NSB, NSEG, ETSG, ETOF, ETMA, ZES3, REP);

OPE étant l'opération d'E/S à exécuter.
NSB étant le numéro de sous-base.
NSEG étant l'adresse virtuelle de l'enregistrement.

ZES3 étant la zone d'E/S du S.E.S.G.
REP étant une zone de transmission d'un code de retour.

D'autre part, le S.E.S.G. est capable de supprimer soit les entrées des tables relatives à un enregistrement d'adresse virtuelle donnée ( appel au sous-programme SUPPRIM), soit une sous-base entière par appel au module SUSB.

Les modules DEM, RESTI, LIBERE, TRUAND, et TRUDEP de gestion de l'espace mémoire sont utilisés indifféremment dans les deux étapes de génération et d'exécution.

Toute la gestion des verrouillages d'enregistrements, de constuction des tables de verrous (TVE) est faite au niveau du noyau du S.E.S.G.

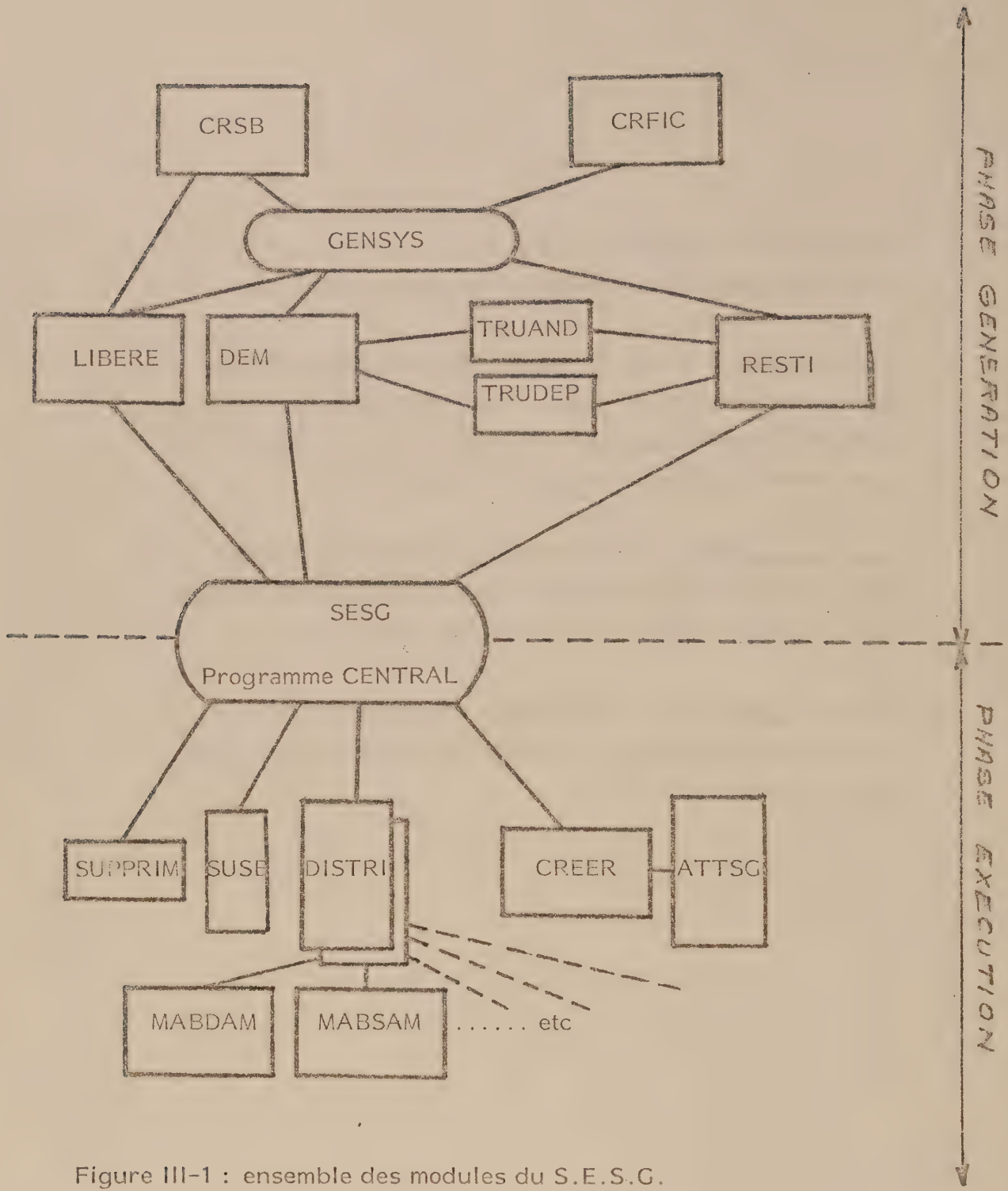



Figure III-1 : ensemble des modules du S.E.S.G.

Il est bon de noter que l'adjonction d'une nouvelle méthode d'accès au système n'affecte que le module DISTRI dans lequel il faut ajouter une nouvelle instruction de CALL au nouveau programme d'E/S. Toutefois, une nouvelle entrée est crée dans la table TMA.

## III-C ORGANIGRAMMES ET LISTINGS DES PROGRAMMES

### III-C-1 Demande de place en mémoire centrale, module DEM

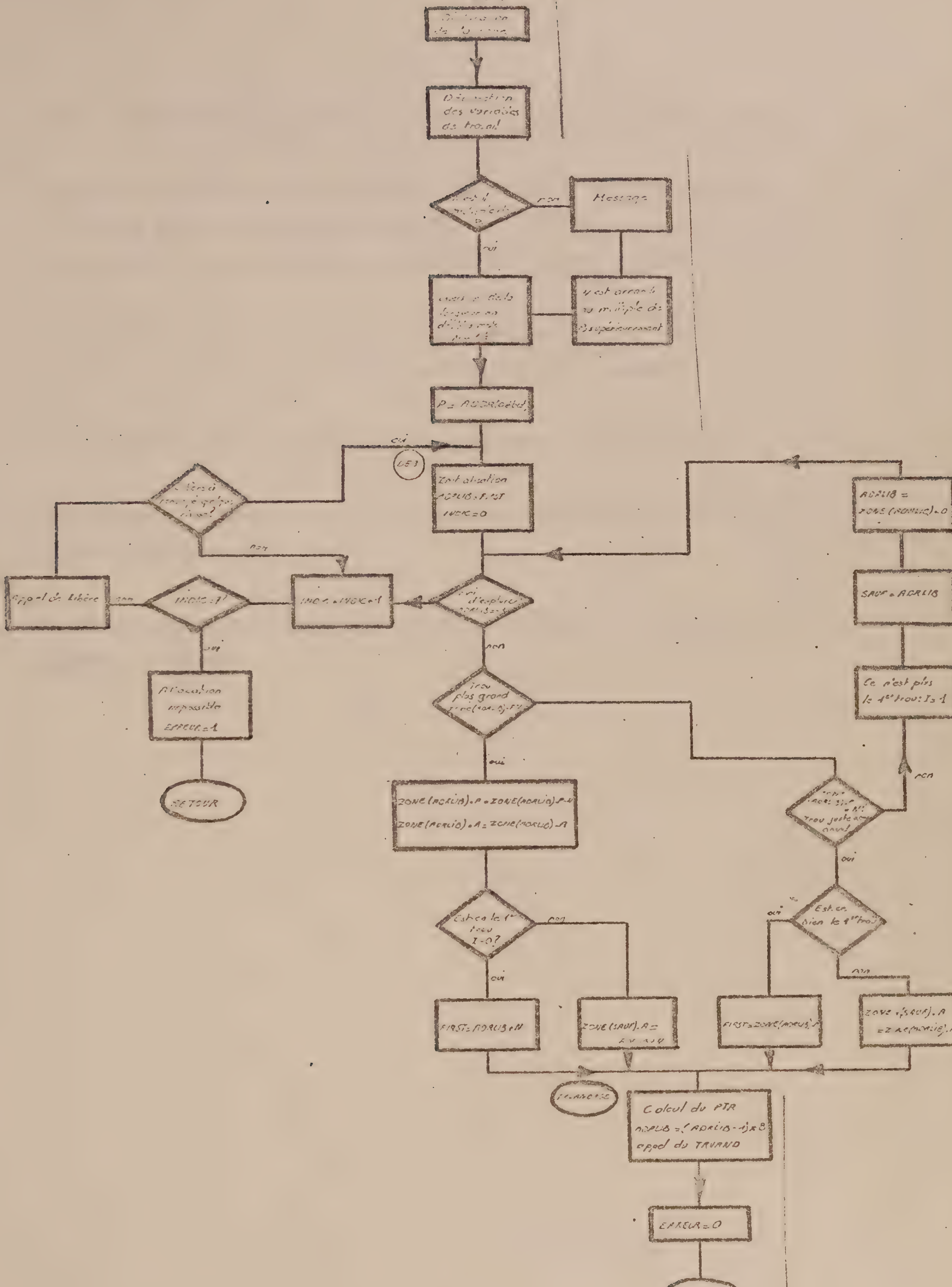
Ce programme cherche suivant un principe semblable à celui de FIRST FIT ( cf II-C-1.2) un "trou" de longueur n octets et fournit l'adresse de ce "trou" au programme appelant.

La zone de travail ZONE est subdivisée en double mots, c'est pourquoi toute demande d'allocation en octets est automatiquement convertie en multiples de 8 octets.

En tête de chaque "trou" de la zone de travail sont stockés une longueur du "trou" et un déplacement vers le "trou" suivant.

Lorsque DEM ne trouve pas de place dans la ZONE, il fait appel au programme LIBERE décrit plus loin.

DEM



Message
non
oui
Initialisation
Allocation impossible
ERREUR = 1
RETOUR
ZONE (ADRLIB).A = ZONE(ADRLIB)-A
SAUF = ADRLIB
Ce n'est plus le 1er trou: I = 1
Est-ce bien le 1er trou?
FIRST = ADRLIB + N
Calcul du PTR
ERREUR = 0
RETOUR

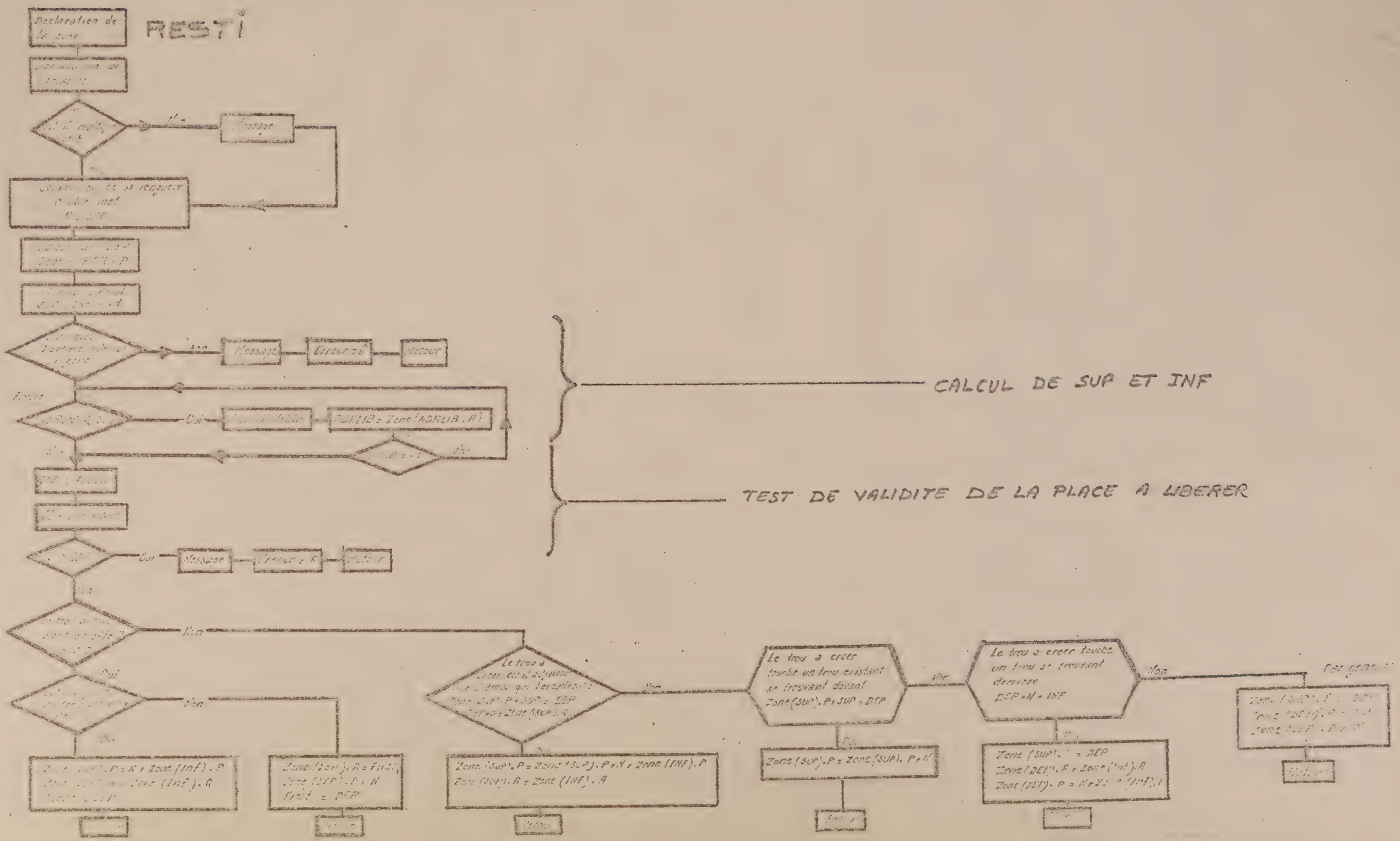
III-C-2 Restitution de place en mémoire centrale : module RESTI :

Le programme RESTI permet de fabriquer un "trou" de longueur N (octets) situé à une adresse donnée.
La longueur N est arrondie au multiple de 8 octets supérieur.

RESTI détermine les "trous" qui encadrent le "trou" à fabriquer, de manière à les fusionner éventuellement avec la nouvelle zone restituée.
Des contrôles de validation sont réalisés dans le programme.

Par exemple, la taille de la zone à restituer ne doit pas déborder de la zone de travail.

Ce module est appelé essentiellement par LIBERE. Il est également appelé par SUSB.


RESTI
Declaration de la zone
Non
Sortie
Oui
Message
Erreur = 6
Retour
Sortie
Oui
ADRLIB = Zone (ADRLIB , N)
Non
Oui
Message
Erreur = 8
Retour
Non
Oui
Non
Zone (SUP) . P = N + Zone (INF) . P
Zone (SUP) . A = Zone (INF) . A
Zone (SUP) . P = P + SUP
Zone (DEP) . P = N
DEP = P
Le trou a creer touche un trou existant se trouvant devant
Zone (SUP) . P + SUP = DEP
Zone (SUP) . P = Zone (SUP) . P + N + Zone (INF) . P
Zone (SUP) . A = Zone (INF) . A
Oui
Non
Zone (SUP) . P = Zone (SUP) . P + N
Le trou a creer touche un trou se trouvant derriere
DEP + N = INF
Oui
Non
Zone (SUP) . P = DEP
Zone (DEP) . P = N + Zone (INF) . P
Zone (DEP) . A = Zone (INF) . A
Cas general
CALCUL DE SUP ET INF
TEST DE VALIDITE DE LA PLACE A LIBERER

```
STT:PROC(N,PTR,ERREUR) ;

    /*****************************************/
    /* CETTE PROCEDURE FABRIQUE UN TROU DE   */
    /* LONGUEUR 'N' (OCTETS) SITUE A         */
    /* L'ADRESSE 'PTR'                       */
    /*****************************************/


LARE

    /*   TETE DE LISTE DES TROUS             */

  FIRST FIXED BIN (31,0) STATIC EXTERNAL,

    /*    ZONE DE TRAVAIL .                  */
    /*    CHAQUE MOT DE CETTE ZONE CONTIENT : */
    /*       - A = INDICE DU TROU SUIVANT    */
    /*       - P = NOMBRE DE MOTS DANS LE TROU */

    1  DEBLT STATIC EXTERNAL,
2 ZONE(30000),
            3  A  FIXED BINARY (31,0),
            3  P  FIXED BINARY (31,0),

    (DEP,SUP,INF,ADRLIB,N,ERREUR) FIXED BINARY (31,0),
    (P,PTR) POINTER;

    /* VALIDATION DE LA LONGUEUR DEMANDEE    */

  ERREUR=0;

  IF N=0 THEN RETURN;
  IF N<0 THEN GO TO CACA;
  IF MOD(N,8)¬=0 THEN DO;
        N=N/8;
        N=8*N+8;
  END;
  N=N/8;

    /* INITIALISATIONS                       */

RREUR=0;
ADDR(DEBUT);
ALL TRUDEP(PTR,P,DEP);
```

```
IF MOD(DEP,8)¬=0 THEN GO TO CACA;
DEP=DEP/8+1;
ADRLIB=FIRST;
SUP,INF=-1;


        /* VALIDATION DE L'ADRESSE                          */

IF (DEP<1)|((DEP+N-1)>3000) THEN DO;
CACA : PUT EDIT('ADRESSE OU LONGUEUR INVALIDE')(SKIP,A);
      ERREUR=5;
      RETURN;
END;


        /* DETERMINATION DES TROUS QUI ENCADRENT    */
        /* LE TROU A FABRIQUER                      */

ENCORE:IF ADRLIB<DEP THEN DO;
      SUP=ADRLIB;
      ADRLIB=ZONE(ADRLIB).A;
      IF ADRLIB¬=-1 THEN GO TO ENCORE;
END;
INF=ADRLIB;
IF    (INF>0  &  DEP+N>INF)
   |  (SUP>0  &  SUP+ZONE(SUP).P>DEP)   THEN GO TO CACA;


        /* CAS OU LA TETE DE LISTE DOIT ETRE        */
        /* MODIFIEE                                 */

IF SUP<0 THEN DO;

        /* CAS OU LE TROU A CREER TOUCHE LE TROU    */
        /* EXISTANT QUI SE TROUVE A LA SUITE        */

IF  (DEP+N)=INF THEN DO;
ZONE (DEP).P=N+ZONE(INF).P;
ZONE(DEP).A=ZONE(INF).A;
FIRST=DEP;
RETURN;
END;

        /* CAS OU LE TROU A CREER NE TOUCHE AUCUN   */
        /* TROU EXISTANT                            */

ZONE(DEP).A=FIRST;
ZONE(DEP).P=N;
FIRST=DEP;
RETURN;
END;


      /* CAS OU LE TROU A CREER TOUCHE DES TROUS    */
       /* EXISTANTS DES DEUX COTES                 */

IF (ZONE(SUP).P+SUP=DEP)&(DEP+N=INF) THEN DO;
```

```
ZONE(SUP).P=ZONE(SUP).P+N+ZONE(INF).P;
ZONE(SUP).A=ZONE(INF).A;
RETURN;
END;


        /* CAS OU LE TROU A CREER TOUCHE UN TROU    */
        /* EXISTANT SE TROUVANT DEVANT              */

IF ZONE(SUP).P+SUP=DEP THEN DO;
ZONE(SUP).P=ZONE(SUP).P+N;
RETURN;
END;


        /* CAS OU LE TROU A CREER TOUCHE UN TROU    */
        /* EXISTANT SE TROUVANT DERRIERE            */

IF DEP +N=INF THEN DO;
ZONE(SUP).A=DEP;
ZONE(DEP).A=ZONE(INF).A;
ZONE(DEP).P=ZONE(INF).P+N;
RETURN;
END;


        /* CAS GENERAL                              */

ZONE(SUP).A=DEP;
ZONE(DEP).A=INF;
ZONE(DEP).P=N;
END TESTI;
```

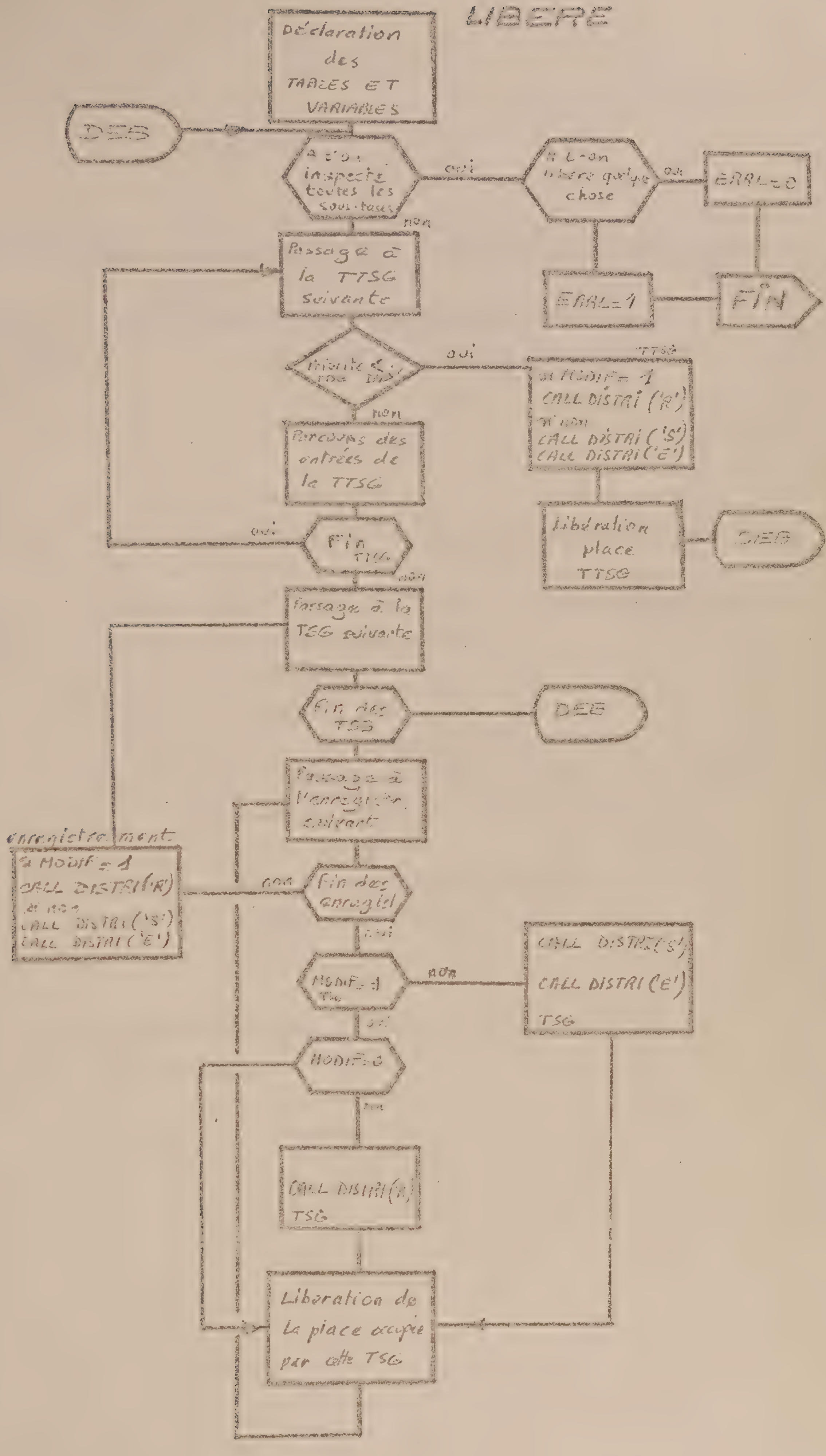
III-C-3 Libération de tables et enregistrements : module LIBERE :

Le programme LIBERE est appelé lorsque le programme DEM a besoin de place supplémentaire qu'il ne trouve pas dans ZONE.
Il est également appelé à la fin de l'exécution du système S.E.S.G.

LIBERE renvoie sur le disque "SBOO" toutes les tables du système dont la priorité réelle est inférieure à INDIC.
LIBERE appelle certains modules de renvoi des enregistrements sur les fichiers adéquats. Ces enregistrements possèdent une priorité inférieure à INDIC.

Les places qui étaient occupées par les enregistrements et tables renvoyés sont restituées par appel au programme RESTI.

LIBERE inspecte toutes les sous-bases existantes et tente de renvoyer l'ensemble des éléments concernés.
Au fur et à mesure que le renvoi s'effectue, le programme LIBERE met à jour les descripteurs associés aux enregistrements transférés.
C'est le sous-programme DISTRI qui choisit la méthode d'accès nécessaire au renvoi des éléments détectés par LIBERE.
LIBERE appelle le programme RECHERCHE qui permet , en fonction de l'adresse virtuelle (NSB, NSEG), de renvoyer les adresses ETOF et ETMA des entrées des tables d'organisation de fichier et table des méthodes d'accès concernées.


LIBERE
Déclaration des TABLES ET VARIABLES
DEB
A t'on inspecté toutes les sous-tâches
oui
non
A t-on libéré quelque chose
oui
ERRL=0
ERRL=1
FIN
Passage à la TTSG suivante
oui
TTSG
si MODIF= 1 CALL DISTRI ('R')
si non CALL DISTRI ('S') CALL DISTRI ('E')
non
Parcours des entrées de la TTSG
Libération place TTSG
DEB
oui
Fin
non
Passage à la TSG suivante
Fin des TSG
DEB
Passage à l'enregistr. suivant
enregistrement
si MODIF= 1 CALL DISTRI('R')
si non CALL DISTRI ('S') CALL DISTRI ('E')
non
Fin des enregist.
oui
MODIF= 1
TSG
non
CALL DISTRI('S')
CALL DISTRI ('E')
TSG
oui
MODIF= 0
non
CALL DISTRI('R')
TSG
Libération de la place occupée par cette TSG

III-C-4 Création de tables du S.E.S.G. : exemple du programme CRSB :

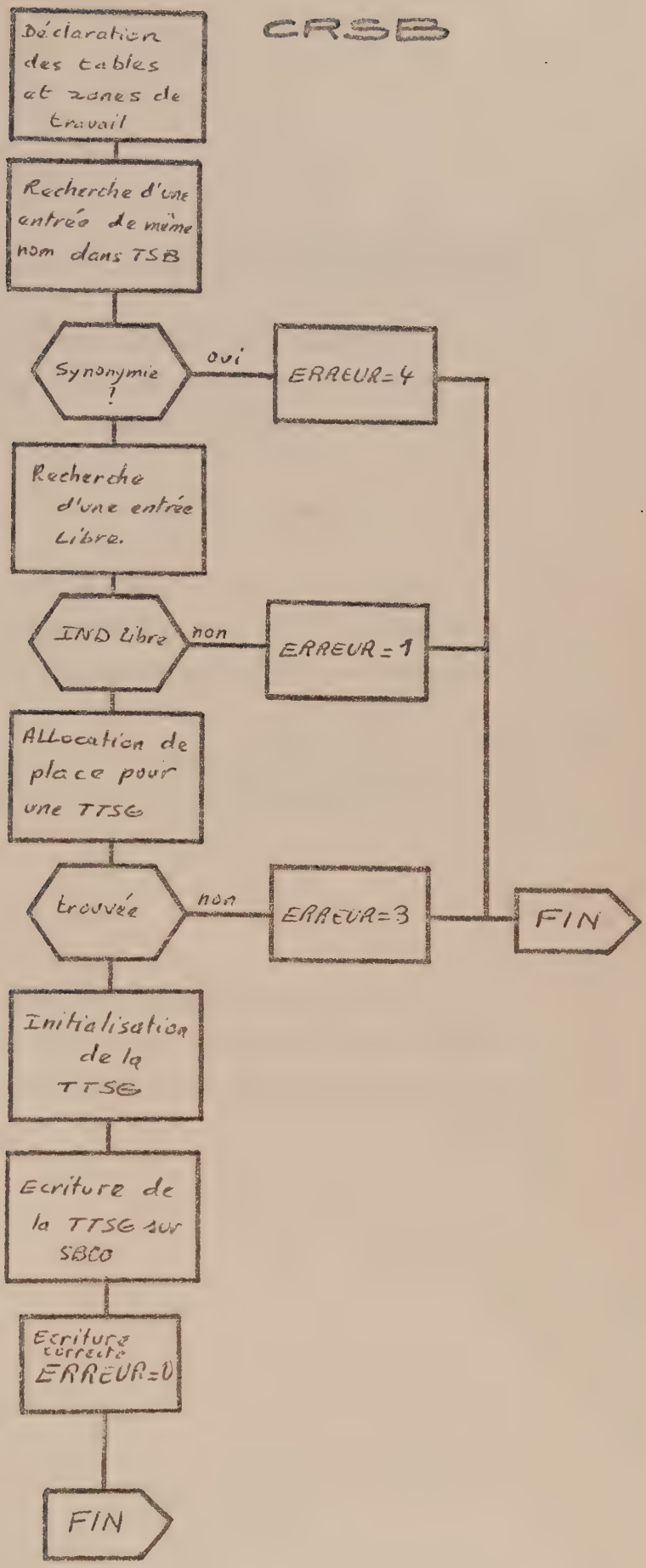
Le programme principal GENSYS fait appel au module CRSB ainsi qu'au module CRFIC afin de créer et d'initialiser certaines tables du système S.E.S.G.
CRSB reçoit les paramètres suivants:

NOM1 : nom de la sous-base à créer
NSB : numéro de l'entrée de la TSB qui sera attribuée par CRSB à cette sous-base
PRTY : priorité qui sera affectée à la TTSG relative à ce nouveau fichier
ERREUR : code de retour, prenant les valeurs:
0, si tout s'est bien passé
1, si la TSB est saturée
2, si erreur d'E/S sur le fichier des tables SB00
3, s'il n'y a plus de place en mémoire centrale
4, si un fichier de même nom NOM1 existe déjà (synonymie)

Ce programme crée donc une table des tables des segments et initialise cette table ainsi que l'entrée associée au niveau de la table des sous-bases.

La création des tables de segments se fait au fur et à mesure des besoins par appel au programme CREER.

CRSB


Déclaration des tables at zones de travail
Recherche d'une entrée de même nom dans TSB
Synonymie ?
oui
ERREUR=4
Recherche d'une entrée Libre.
IND Libre
non
ERREUR=1
Allocation de place pour une TTSG
trouvée
non
ERREUR=3
FIN
Initialisation de la TTSG
Ecriture de la TTSG sur SBCO
Ecriture correcte
ERREUR=0
FIN

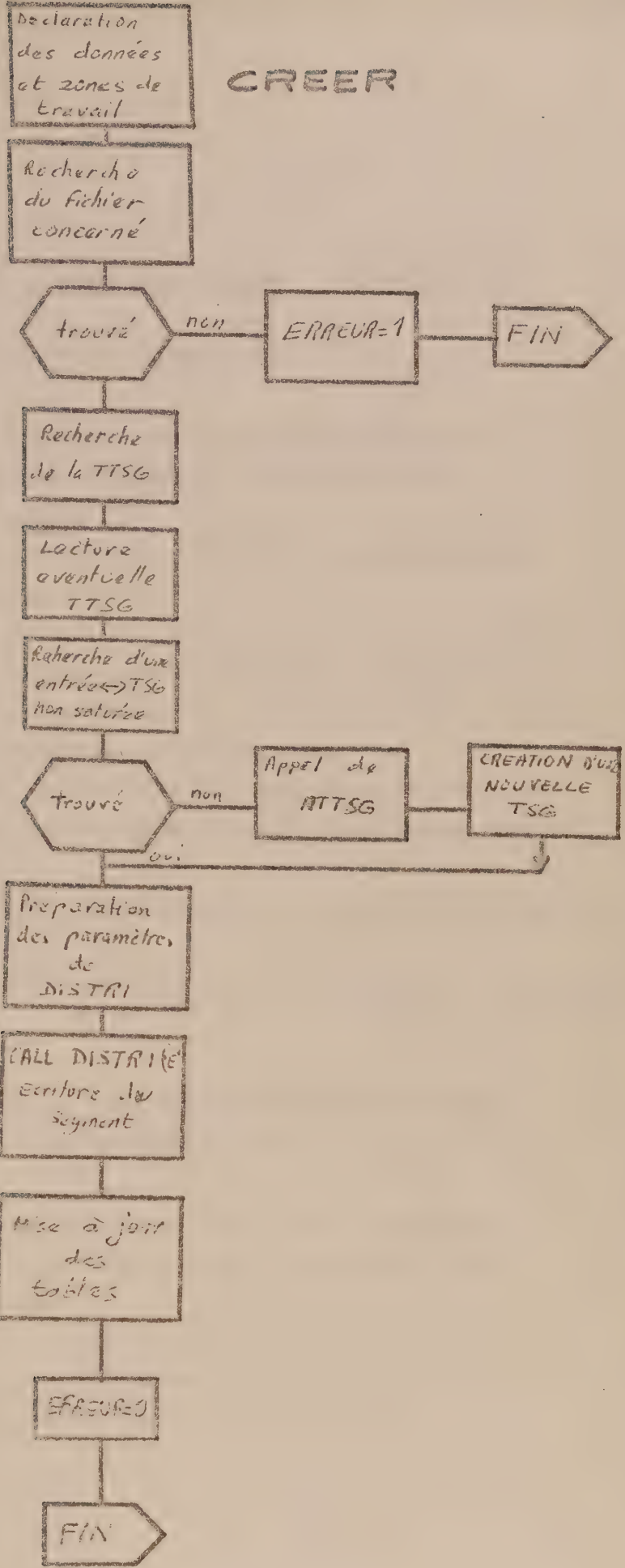
III-C-5 Ajout d'un enregistrement dans un fichier : programme CREER

Ce module CREER est appelé par le programme central du S.E.S.G. chaque fois qu'il est nécessaire de créer dans un fichier donné un enregistrement implanté à une adresse mémoire centrale donnée.

CREER réalise des contrôles avant de lire (éventuellement) les tables du S.E.S.G.
Dans le cas où il n'existe plus d'entrée libre à associer au nouvel enregistrement, CREER active la procédure interne ATTSG qui permet "d'allonger" la table des tables de segments relative à la sous-base.

Le transfert de la mémoire centrale vers le support physique se fait par appel à la procédure DISTRI qui se charge d'effectuer cette opération.
Les paramètres transmis sont:

NOM1 : nom du fichier concerné
NSEG : numéro adresse virtuelle de l'enregistrement créé
ER : adresse mémoire centrale où est implanté l'enregistrement
PRTY : priorité accordée à l'enregistrement nécessaire à la gestion de l'espace mémoire
ERREUR : code de retour


CREER
Declaration des données et zones de travail
Recherche du fichier concerné
trouvé
non
ERREUR=1
FIN
Recherche de la TTSG
Lecture eventuelle TTSG
Recherche d'une entrée ↔ TSG non saturée
trouvé
non
Appel de ATTSG
CREATION D'UNE NOUVELLE TSG
oui
Preparation des paramètres de DISTRI
CALL DISTRI (Ecriture du segment
Mise à jour des tables
ERREUR=0
FIN

III-C-6 Choix de la technique d'accès aux données : module DISTRI

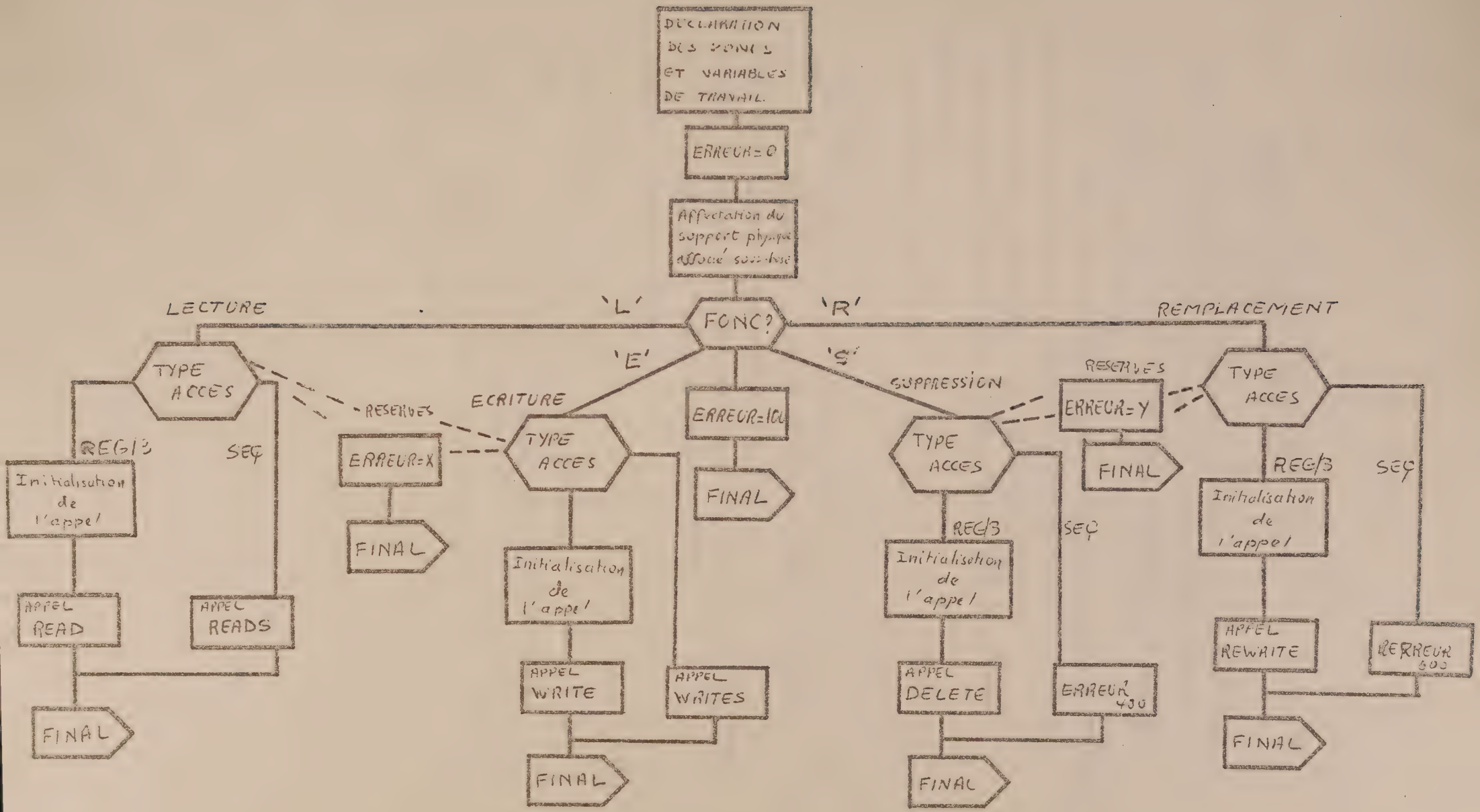
C'est un des points les plus importants du système S.E.S.G.

Le regroupement de tous les appels aux modules d'E/S au niveau de DISTRI simplifie énormément toute modification ou reconsidération des techniques d'accès aux données.
La procédure DISTRI est appelée par un certain nombre de programmes du S.E.S.G. avec les paramètres suivants:

FONC : type de la fonction élémentaire à exécuter
NSB : numéro de la sous-base concernée
NSEG : numéro relatif du segment ou enregistrement
PTSG : adresse réelle de l'entrée de la table des segments concernée
PTOF : adresse réelle de l'entrée de la table des organisations relative au fichier traité
PTHA : adresse réelle de l'entrée de la table des méthodes d'accès relative à l'opération
ZES3 : adresse mémoire de la zone d'émission-réception de l'enregistrement
ERREUR : code de retour

L'organigramme suivant montre bien les différents aiguillages d'appel des modules d'E/S choisis par le programme appelant.

Toutes les étiquettes dont le qualificateur commence par les caractères 'RESERV' sont prévues pour les futurs appels aux programmes d'E/S qui seront éventuellement ajoutés.
Avant de faire appel à DISTRI, le S.E.S.G. vérifie au préalable si l'enregistrement à lire n'est pas en mémoire ou si l'enregistrement à réécrire n'a pas été modifié auxquels cas l'appel ne se fera pas.


DECLARATION DES ZONES ET VARIABLES DE TRAVAIL.
ERREUR=0
Affectation du support physique associé sous-tâche
FONC?
'L'
'E'
'S'
'R'
LECTURE
ECRITURE
SUPPRESSION
REMPLACEMENT
TYPE ACCES
REG/S
SEQ
Initialisation de l'appel
APPEL READ
APPEL READS
FINAL
RESERVES
ERREUR=X
ERREUR=100
TYPE ACCES
Initialisation de l'appel
APPEL WRITE
APPEL WRITES
FINAL
TYPE ACCES
REG/S
SEQ
Initialisation de l'appel
APPEL DELETE
ERREUR 400
FINAL
RESERVES
ERREUR=Y
FINAL
TYPE ACCES
REG/S
SEQ
Initialisation de l'appel
APPEL REWRITE
ERREUR 400
FINAL

III-C-7 Exemples de programmes d'E/S

Nous avons choisi comme exemple un ensemble de quatre programmes d'E/S en REGIONAL/3 (en réalité quatre points d'entrées différents dans le même module), REGIONAL/3 étant une méthode d'accès direct par piste et clé dans cette piste.
Ces programmes excessivement simples (READ, WRITE, DELETE, REWRITE) sont activés pour toutes les E/S des systèmes de gestion de bases de données utilisant cette méthode d'accès sous le S.E.S.G.
Toute nouvelle méthode d'accès entraîne l'écriture de programmes de ce type:
Les paramètres transmis, dans ce cas, par la procédure DISTRI sont:

Pour le module READ:

| | |
|---|---|
| NSB | : numéro de la sous-base concernée |
| PP | : adresse de la zone de réception en mémoire centrale |
| CLEF | : clé de l'enregistrement à lire |
| REGION | : numéro de la piste susceptible de contenir l'enregistrement |
| M | : longueur en octets de l'enregistrement |
| ERREUR | : code de retour |

Pour le module WRITE:

| | |
|---|---|
| NSB | : numéro de la sous-base concernée |
| PP | : adresse de la zone d'émission en mémoire centrale |
| CLEF | : clé de l'enregistrement à écrire |
| REGION | : numéro de la piste de réception |
| ERREUR | : code de retour |

Pour le module DELETE:

NSB : numéro de la sous-base concernée
CLEF : clé de l'enregistrement à supprimer
REGION : numéro de la piste contenant l'enregistrement
ERREUR : code de retour

Pour le module REWRITE:

NSB : numéro de la sous-base concernée
PP : adresse de la zone de réception en mémoire centrale
CLEF : clé de l'enregistrement à modifier
REGION : numéro de la piste contenant l'enregistrement ancien
ERREUR : code de retour

Comme nous pouvons le constater dans les listings suivants, nous nous sommes limités volontairement à cinq fichiers différents (sous-base SB01, SB02, SB03, SB04 et SB05). Il est très facile d'étendre ces ordres d'E/S à n fichiers.

III-D Exemples d'interrogation de bases de données :

Nous avons créé une base de données EXEMPLE constituée de deux sous-bases ACCIDENT et RECENSEMENT.
La structuration ainsi que la définition interne de cette base de données sont entièrement assurées par le système GRISBI (commande $DEFARTS, $DEFARTB, $CODE, $CREREF etc...).
La création des tables de gestion d'E/S du S.E.S.G. a été réalisée à partir des données suivantes:

```
NOMBASE = EXEMPLE
TOTAL-ENR-LOGIQUES = 10000
        x NOMFICH = ACCIDENT
        LIMITES = 0,5000
        SUPPORT = 3330
        MET-ACCES = REG/3
        MOD-ACCES = 1
        x NOMFICH = RECENSEMENT
        LIMITES = 5000, 10000
        SUPPORT = 2314
        MET-ACCES = SEQUENCE
        MOD-ACCES = 2
        x NOMFICH = SYSOUT, SORTIE
```

Les deux sous-bases ont été chargées correctement. Pour bien comprendre les résultats donnés dans les listings suivants, il est bon d'expliquer les commandes d'interrogation du système GRISBI utilisées.

Les conditions apparaissant dans la définition des ensembles d'enregistrements peuvent prendre des formes diverses utilisant des opérateurs du type ET, OU, =, ¬ =, >, > =, <, < =, ENTRE, TOTAL, POUR NOMBRE, MIN, MAX, MOY, SOM, EC ...

Les commandes $RECHERCHER sont suivies d'une ou plusieurs commandes $SORTIR qui permettent de préciser les informations à extraire d'un ensemble d'enregistrements (définies dans les commandes $RECHERCHER précédentes) ainsi que les calculs à effectuer sur ces informations.

Le format de $SORTIR est le suivant:

$SORTIR NOM = SYSOUT E1 (expression, .....,expression)

E2 (expression, ....., expression) ...En (expression, ....., expression)

Le paramètre NOM = SYSOUT indique que les sorties doivent se faire sur imprimante. SYSOUT a été défini au niveau du S.E.S.G. comme fichier de sortie.
Dans la prochaine version, le paramètre NOM = SYSOUT pourra être remplacé par NOM = XXXXXX pour obtenir les résultats sur un fichier dont on précisera le DDname.

```
&RECHERCHER ACCIDENT
 GENE:(DEP>='01' ET DEP<='19')
   PARMI GENE
   PAR1:(ORGANISME='GENDARME')
   PAR2:(NOMBRE(PIETON)>=1)
     PARMI PAR2
     PAR3:(NOMBRE(PASSAGER)>=1)
     FIN PAR2
 FIN GENE
&SORTIE   NOM=SYSOUT
 GENE(NOMBRE PASSAGER,NOMBRE PIETON,CTRL,INTEMP,LUMIERE)
 PAR1(CONDUCTEUR.AGE,MAX CONDUCTEUR.AGE,MIN CONDUCTEUR.AGE)
 PAR1(RAPPORT GENE)
 PAR3(RAPPORT PAR2,MOYG PIETON.AGE,MAXG PIETON.AGE,MING PIETON.AGE)
 PAR2(NOMBRE PASSAGER POUR CHAQUE VEHICULE, NOMBRE PASSAGER)
 &FIN
```

commande de recherche relative au "fichier" accident

ensembles de conditions définissant des sous-ensembles d'enregistrements.

Commande de sortie des informations définies dans la commande $RECHERCHER précédente

Commande $FIN lançant la recherche et la sortie des résultats demandés précédemment.

SYSTEME GRISBI
E/S en REGIONAL/3

*EXEMPLE DE SORTIE SUR IMPRIMANTE*

```
                    ****************************************
                    * RECHERCHE SUR LA SOUS-BASE ACCIDENT  *
                    ****************************************

     GENE : ACCIDENT     AYANT:
( LIEU.DEP >= 'C1       ' ET LIEU.DEP <= '19      ' )

          NOMBRE D'ENREGISTREMENTS APPARTENANT A CETTE ETIQUETTE =  19


***LUMIERE=PLEIN JO

***INTEMP=0

***CTRL=NON

    ***NOMBRE DE
          PIETON       = 0

    ***NOMBRE DE
          VEHICULE.PASSAGER     = 0

***LUMIERE=PLEIN JO

***INTEMP=0

***CTRL=NON

    ***NOMBRE DE
          PIETON       = 0

    ***NOMBRE DE
          VEHICULE.PASSAGER     = 1

***LUMIERE=PACLAIR
```

```
      ***NOMBRE DE
            PIETON      = 0

      ***NOMBRE DE
            VEHICULE.PASSAGER      = 3


***LUMIERE=PLEIN JO


***INTEMP=PLUIE LE

      ***NOMBRE DE
            PIETON      = 0

      ***NOMBRE DE
            VEHICULE.PASSAGER      = 4


***LUMIERE=PACLAIR


***INTEMP=C

      ***NOMBRE DE
            PIETON      = C

      ***NOMBRE DE
            VEHICULE.PASSAGER      = 5


***LUMIERE=AUBE


***INTEMP=C


***CTRL=ACA

      ***NOMBRE DE
            PIETON      = 0

      ***NOMBRE DE
            VEHICULE.PASSAGER      = 2


***LUMIERE=PLEIN JC


***INTEMP=C
```

PIETON = 0

***NOMBRE DE
VEHICULE.PASSAGER = 0

***LUMIERE=PACLAIR

***INTEMP=PLUIE LE

***NOMBRE DE
PIETON = 2

***NOMBRE DE
VEHICULE.PASSAGER = 0

***LUMIERE=NUINCR

***INTEMP=0

***NOMBRE DE
PIETON = 0

***NOMBRE DE
VEHICULE.PASSAGER = 0

***LUMIERE=PLEIN JO

***INTEMP=0

***NOMBRE DE
PIETON = 1

***NOMBRE DE
VEHICULE.PASSAGER = 1

***LUMIERE=PLEIN JO

***INTEMP=0

***NOMBRE DE
PIETON = 1

```
***LUMIERE=AUBE


***INTEMP=0

     ***NOMBRE DE
          PIETON      = 0

     ***NOMBRE DE
          VEHICULE.PASSAGER      = 1


***LUMIERE=PACLAIR


***INTEMP=0

     ***NOMBRE DE
          PIETON      = 0

     ***NOMBRE DE
          VEHICULE.PASSAGER      = 2


***LUMIERE=PLEIN JO


***INTEMP=0

     ***NOMBRE DE
          PIETON      = 0

     ***NOMBRE DE
          VEHICULE.PASSAGER      = 0


***LUMIERE=INSUFFIS


***INTEMP=0

     ***NOMBRE DE
          PIETON      = 0

     ***NOMBRE DE
          VEHICULE.PASSAGER      = 2


***LUMIERE=PLEIN JO
```

***INTEMP=J

***NOMBRE DE
PIETON = 1

***NOMBRE DE
VEHICULE.PASSAGER = 1

***LUMIERE=PLEIN JO

***INTEMP=C

***CTRL=NON

***NOMBRE DE
PIETON = 0

***NOMBRE DE
VEHICULE.PASSAGER = 0

***LUMIERE=AUBE

***INTEMP=C

***CTRL=NON

***NOMBRE DE
PIETON = 0

***NOMBRE DE
VEHICULE.PASSAGER = 1

***LUMIERE=CLAIR

***INTEMP=C

***NOMBRE DE
PIETON = 0

***NOMBRE DE
VEHICULE.PASSAGER = 1

PARMI CELLE PARI : ACCIDENT AYANT:
( ORGANISME = 'GENDARME' )

NOMBRE D'ENREGISTREMENTS APPARTENANT A CETTE ETIQUETTE = 19

RAPPORT DE PARI A CENE

*** = 1.000

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 47

***NOMBRE DE REALISATIONS CONCERNEES = 2

***MAXIMUM DE
VEHICULE.CONDUCTEUR.AGE = 50

***NOMBRE DE REALISATIONS CONCERNEES = 2

VEHICULE (2)

***AGE=50

VEHICULE (1)

***AGE=47

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 54

***NOMBRE DE REALISATIONS CONCERNEES = 2

***MAXIMUM DE
VEHICULE.CONDUCTEUR.AGE = 66

***NOMBRE DE REALISATIONS CONCERNEES = 2

VEHICULE (2)

***AGE=66

VEHICULE (1)

***AGE=54

VEHICULE (2)

***AGE=42

VEHICULE (1)

***AGE=36

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 25

***NOMBRE DE REALISATIONS CONCERNEES = 2

***MAXIMUM DE
VEHICULE.CONDUCTEUR.AGE = 41

***NOMBRE DE REALISATIONS CONCERNEES = 2

VEHICULE (2)

***AGE=41

VEHICULE (1)

***AGE=25

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 17

***NOMBRE DE REALISATIONS CONCERNEES = 2

***MAXIMUM DE
VEHICULE.CONDUCTEUR.AGE = 39

***NOMBRE DE REALISATIONS CONCERNEES = 2

VEHICULE (2)

***AGE=39

VEHICULE (1)

***AGE=17

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 17

```
          ***NOMBRE DE REALISATIONS CONCERNEES = 1

      ***MAXIMUM DE
            VEHICULE.CONDUCTEUR.AGE     = 22

          ***NOMBRE DE REALISATIONS CONCERNEES = 1


  VEHICULE (1)

***AGE=22

      ***MINIMUM DE
            VEHICULE.CONDUCTEUR.AGE     = 00

          ***NOMBRE DE REALISATIONS CONCERNEES = 2

      ***MAXIMUM DE
            VEHICULE.CONDUCTEUR.AGE     = 32

          ***NOMBRE DE REALISATIONS CONCERNEES = 2


  VEHICULE (2)

***AGE=32


  VEHICULE (1)

***AGE=00

      ***MINIMUM DE
            VEHICULE.CONDUCTEUR.AGE     = 37

          ***NOMBRE DE REALISATIONS CONCERNEES = 1

      ***MAXIMUM DE
            VEHICULE.CONDUCTEUR.AGE     = 37

          ***NOMBRE DE REALISATIONS CONCERNEES = 1


  VEHICULE (1)

***AGE=37

      ***MINIMUM DE
            VEHICULE.CONDUCTEUR.AGE     = 52
```

***MAXIMUM DE
VEHICULE.CONDUCTEUR.AGE = 52

***NOMBRE DE REALISATIONS CONCERNEES = 1

VEHICULE (1)

***AGE=52

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 50

***NOMBRE DE REALISATIONS CONCERNEES = 2

***MAXIMUM DE
VEHICULE.CONDUCTEUR.AGE = 54

***NOMBRE DE REALISATIONS CONCERNEES = 2

VEHICULE (2)

***AGE=54

VEHICULE (1)

***AGE=50

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 20

***NOMBRE DE REALISATIONS CONCERNEES = 1

***MAXIMUM DE
VEHICULE.CONDUCTEUR.AGE = 20

***NOMBRE DE REALISATIONS CONCERNEES = 1

VEHICULE (1)

***AGE=20

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 17

***NOMBRE DE REALISATIONS CONCERNEES = 1

VEHICULE.CONDUCTEUR.AGE = 27

***NOMBRE DE REALISATIONS CONCERNEES = 1

VEHICULE (1)

***AGE=27

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 28

***NOMBRE DE REALISATIONS CONCERNEES = 2

***MAXIMUM DE
VEHICULE.CONDUCTEUR.AGE = 53

***NOMBRE DE REALISATIONS CONCERNEES = 2

VEHICULE (2)

***AGE=53

VEHICULE (1)

***AGE=28

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 26

***NOMBRE DE REALISATIONS CONCERNEES = 1

***MAXIMUM DE
VEHICULE.CONDUCTEUR.AGE = 26

***NOMBRE DE REALISATIONS CONCERNEES = 1

VEHICULE (1)

***AGE=26

***MINIMUM DE
VEHICULE.CONDUCTEUR.AGE = 22

***NOMBRE DE REALISATIONS CONCERNEES = 2

```
      ***NOMBRE DE REALISATIONS CONCERNEES = 2


  VEHICULE (2)
***AGE=22


  VEHICULE (1)
***AGE=54
     ***MINIMUM DE
           VEHICULE.CONDUCTEUR.AGE      = 57

        ***NOMBRE DE REALISATIONS CONCERNEES = 2

     ***MAXIMUM DE
           VEHICULE.CONDUCTEUR.AGE      = 59

        ***NOMBRE DE REALISATIONS CONCERNEES = 2


  VEHICULE (2)
***AGE=59


  VEHICULE (1)
***AGE=57
     ***MINIMUM DE
           VEHICULE.CONDUCTEUR.AGE      = 19

        ***NOMBRE DE REALISATIONS CONCERNEES = 1

     ***MAXIMUM DE
           VEHICULE.CONDUCTEUR.AGE      = 19

        ***NOMBRE DE REALISATIONS CONCERNEES = 1


  VEHICULE (1)
***AGE=19
```

[illegible] : ACCIDENT [illegible]:
( NOMBRE PIETON ) >= 1 ;

NOMBRE D'ENREGISTREMENTS APPARTENANT A CETTE ETIQUETTE = 4

***NOMBRE GENERAL DE
VEHICULE.PASSAGER [illegible] = 2

***NOMBRE DE
VEHICULE.PASSAGER

POUR CHAQUE VEHICULE

***VEHICULE (1) = 0

***NOMBRE DE
VEHICULE.PASSAGER

POUR CHAQUE VEHICULE

***VEHICULE (1) = 1

***NOMBRE DE
VEHICULE.PASSAGER

POUR CHAQUE VEHICULE

***VEHICULE (1) = 0

***NOMBRE DE
VEHICULE.PASSAGER

POUR CHAQUE VEHICULE

***VEHICULE (1) = 1

PARMI PAR2 PAR3 : ACCIDENT AYANT:
( NOMBRE( VEHICULE.PASSAGER ) >= 1 )

NOMBRE D'ENREGISTREMENTS APPARTENANT A CETTE ETIQUETTE = 2

***MINIMUM GENERAL DE
PIETON.AGE = 07

***NOMBRE DE REALISATIONS CONCERNEES = 2

***MAXIMUM GENERAL DE
PIETON.AGE = 81

***NOMBRE DE REALISATIONS CONCERNEES = 2

***MOYENNE GENERALE DE
PIETON.AGE = 44.000

***NOMBRE DE REALISATIONS CONCERNEES = 2

RAPPORT DE PAR3 A PAR2

*** = 0.500

# CONCLUSION

La recherche dans le domaine de la conception des systèmes de gestion de bases de données est de plus en plus importante, et la construction de ces derniers reste très complexe.
Plusieurs méthodes de conception et de réalisation font l'objet de nombreuses publications mais demeurent limitées à des problèmes particuliers dans la majorité des cas.
Ces systèmes de gestion ont donné lieu, ces derniers temps, à des critiques sévères qui ne sont pas à notre avis totalement dénuées de fondements.
En effet, si nous considérons qu'ils comportent les deux aspects suivants:

- l'aspect "interprétation - exécution des fonctions" et
- l'aspect "accès aux données - gestion des supports physiques"

nous nous apercevons que ces systèmes se développent mais que cette évolution concerne surtout le premier aspect.

Le probème de l'accès aux données semble pourtant être d'une importance capitale et des solutions non satisfaisantes entraînent obligatoirement un retard dans l'évolution des S.G.B.D. par rapport à l'évolution des logiciels et matériels de base.
Ce travail, qui est le fruit d'une expérience générale sur quelques systèmes de gestion de bases de données représentatifs (IMS, TOTAL, MIISFIIT, SOCRATE) et d'une expérience particulière dans le cadre du système GRISBI, introduit un certain nombre d'idées originales.

1- L'illusion d'utilisation d'un seul type d'accès par les S.G.B.D. Le changement de la valeur d'un seul paramètre donne la possibilité d'aborder et d'essayer toutes les méthodes d'accès existantes.

2- La possibilité donnée aux S.G.B.D. de profiter automatiquement à travers le S.E.S.G., sans bouleversement, de certaines évolutions du logiciel et du matériel de base.

# BIBLIOGRAPHIE

BIBLIOGRAPHIE

(1) J. ARSAC "Les systèmes de conduite des ordinateurs" DUNOD 1968.

(2) CODASYL SYSTEMS COMMITTEE "A survey of generalized data base management systems" mai 1969 A.C.M.

(3) CODASYL DATA BASE TASK GROUP
April 1971 A.C.M.
(disponible à BCS-Londres et IFIP-Amsterdam)

(4) R.C. DALEY et P.G. NEUMAN "A generalized purpose file system for secondary storage" proceedings de la FALL JOINT COMPUTER CONFERENCE 27 SPARTAN BOOKS Washington D.C., 1965.

(5) L'exploitation partagée des calculateurs AFIRO
J. BERTIN, M. RITOUT, J-C. ROUGIER - DUNOD 1967.

(6) P. DENNING "Virtual memory" A.C.M. computing surveys Vol. 2 numéro 3 Septembre 1970.

(7) E.W. DIJKSTRA "Co-operating sequential processes" dans "programming languages" édités par F. GENUYS chez ACADEMIC PRESS en 1968.

(8) E.W. DIJKSTRA " The structure of THE-multiprogramming system" A.C.M. communications Vol. 11 Numéro 5 Mai 1968.

(9) "The future is now" Harward Business Review NOLAN.

(10) T.W. OLLE Introduction to "Feature analysis of generalized data base management systems" A.C.M. communications Vol. 14 Numéro 5 Mai 1971.

(11) Les Banques de Données - CENTI
Alain GIBERT

(12) Operating systems techniques proceedings of Seminar held at Queen's University Belfast 1971, edited by C.A.R. HOARE and R.H. PEROTT.

(13) Les techniques de l'informatique - Congrès AFCET 1972
Volume 1, 2.